Chapter 4

Interpolation and Numerical Differentiation

(插值法與數值微分法)

§4.0 簡介 (Introduction)

在科學和工程上, 我們常碰到有一組實驗資料需要求一函數來表示. 此即爲插值 (interpolation) 之問題. 而最終目的 (goal) 是 **to determine the values at intermediate points, to approximate the integral or derivative** of the underlying function, or to give a smooth or continuous representation of the variables in the problem.

更一般性地我們提出三種問題:

First: 給一組數值資料–如下列函數值表:

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

能否求得一簡單公式 (函數) 正好通過這些點?(即求其函數之表示法).

Second: 類似問題一, 但所給之資料包含誤差 (errors) (即眞值被 errors 所污染或 data 可能來自 a physical expeniment). 欲求一公式表示這些 data (approximately), 若可能的話, 過濾掉 (filter out) the errors. 此種問題即

是所謂 **data fitting**.

Third: 有一函數 f, 也許是一個 computer procedure 之型式, 但若要去計算它是相當昂貴的. 因此我們尋求一函數 g, 其較易於計算且能產生 f 的合理近似值 (即用函數 g 來近似函數 f).

★ 所有這些問題都是欲求一簡單函數 p 來代表或近似所給的 data 或函數 f. 此簡單函數 p 有很多種型式, 最普通的是多項式函數. 只要此函數 p 得到, 則它可以取代原函數 f 被使用, 例如求 f 的函數值或積分或導數等.

★ 對於這些問題的不同型式之解法有: 多項式函數插值法 (**interpolation**), 仿樣函數 (**the Spline functions**)(以一群函數之線性組合的方法), 和最小平方法 (**the method of least squares**).

定理 **1.** *(Weierstrass Approximation Theorem) Suppose that f is well-defined and cotinuous on $[a,b]$. For each $\varepsilon > 0$, there exists a polynomial $P(x)$ defined on $[a,b]$, with the property that*

$$|f(x) - P(x)| < \varepsilon, \quad \textit{for all } x \in [a,b].$$

★ The Taylor polynomials agree as closely as possible with the given function at a specific point, but they concentrate **their accuracy only near that point.** A good interpolation polynomial needs to provide a relatively accurate approximation over an entire interval, and Taylor polynomials do not do that.

For example, the Taylor polynomials of $f(x) = e^x$ about $x_0 = 0$ is

$$P_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}$$

4.1 Polynomial Interpolation

已知有 $n+1$ 個相異點如下表:

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

代表歐氏平面 (Cartesian plane)上的 $n+1$ 個點. 欲求一多項式通過所有這些點, 即求一多項式 p 滿足

$$p(x_i) = y_i, \quad 0 \le i \le n. \tag{4.1}$$

稱此多項式插入此表 (p interpolates the table). 點 x_i 稱爲結點 (nodes), 多項式 p 之次數 (degree) 依據點的個數來決定.

首先考慮最簡單之情形: $n=0$, 即只經過一點(x_0, y_0), 則常數函數 (或零次函數) $p(x_0) = y_0$ 即滿足此要求. 其次當 $n=1$:

x	x_0	x_1
y	y_0	y_1

因過兩點決定一直線, 故線性函數

$$\begin{aligned} p(x) &= \left(\frac{x-x_1}{x_0-x_1}\right) y_0 + \left(\frac{x-x_0}{x_1-x_0}\right) y_1 \\ &= y_0 + \left(\frac{y_1-y_0}{x_1-x_0}\right)(x-x_0) \end{aligned}$$

即滿足此問題之解. 即直線 $p(x)$ 滿足 $p(x_0)=y_0$, 且 $p(x_1)=y_1$. 這"一次函數" p 被用來當線性插入法(linear interpolation).

例題 **1.** 求一次數最低之多項式插入下表:

x	*1.4*	*1.25*
y	*3.7*	*3.9*

An interpolating polynomial 可寫成各種不同型式, 較有名的有:

1. Newton form—— 最爲方便和有效應用於 computer 之計算.

2. Lagrange form—— 觀念上易了解, 易於手算.

Interpolating Polynomial: Lagrange form

Suppose 欲 interpolate 任一函數經過 $n+1$ 個相異點 $x_0, x_1, x_2, ..., x_n$, 首先定義 $n+1$ 個次數 (degree) 爲 n 之特殊多項式, $\ell_0, \ell_1, ..., \ell_n$, 稱它們爲 基本函數 **(Cardinal functions)**. 使其具有此性質:

$$\ell_i(x_j) = \delta_{ij} = \begin{cases} 0, & \text{if} \quad i \neq j \\ 1, & \text{if} \quad i = j \end{cases}$$

這些基本函數定義爲:

$$\begin{aligned} \ell_i(x) &= \prod_{\substack{j=0 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right), \quad 0 \leq i \leq n & (4.2) \\ &= \left(\frac{x - x_0}{x_i - x_0} \right) \left(\frac{x - x_1}{x_i - x_1} \right) \cdots \left(\frac{x - x_{i-1}}{x_i - x_{i-1}} \right) \left(\frac{x - x_{i+1}}{x_i - x_{i+1}} \right) \cdots \left(\frac{x - x_n}{x_i - x_n} \right) & (4.3) \end{aligned}$$

Clearly,

$$\begin{cases} \ell_i(x_i) = 1, & \forall\, i = 0, 1, 2, ..., n \\ \ell_i(x_j) = 0, & \forall\, j \neq i \end{cases}$$

當這些函數已建立, 則插入任一函數 f, 其經過所給的 $n+1$ 個相異點, 之插值多項式的**Lagrange form**爲

$$p_n(x) = \sum_{i=0}^{n} \ell_i(x) f(x_i) \tag{4.4}$$

即多項式 p_n 是 $\ell_i(0 \le i \le n)$ 之線性組合.

例題 **2.** 寫出 *the* **Cardinal functions (基本函數)**, 適合於 *interpotating the following table* 且求出 *the Lagrange form of the interpolating polynomial.*

x	$1/3$	$1/4$	1
$f(x)$	2	-1	7

解答 :

因此, 插值多項式爲

$$\begin{aligned} p_2(x) &= \sum_{i=0}^{2} \ell_i(x) f(x_i) \\ &= -36\left(x-\frac{1}{4}\right)(x-1) - 16\left(x-\frac{1}{3}\right)(x-1) + 14\left(x-\frac{1}{3}\right)\left(x-\frac{1}{4}\right) \\ &= 2 + \left(x-\frac{1}{3}\right)\left[36 + \left(x-\frac{1}{4}\right)(-38)\right]. \end{aligned}$$

上式中第一式即爲**Lagrange form**.

Polynomial Interpolation by Neville's Algorithm

由所給之數値表:

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

求 a polynomial interpolant 的另一種方法稱爲Neville's Algorithm.

設$p_{a,b,c,\dots,s}(x)$ 是依所給之值中結點 $x_a, x_b, \dots, x_s$ 所得之插值多項式. 開始以常數多項式 $p_i(x) = f(x_i)$. 選擇兩個 Nodes x_i and x_j with $i > j$ 定義一遞回公式:

Recursively Generated Lagrange Polynomials

$$p_{u,\dots,v}(x) = \left(\frac{x - x_j}{x_i - x_j}\right) p_{u,\dots,j-1,j+1,\dots,v}(x) - \left(\frac{x - x_i}{x_i - x_j}\right) p_{u,\dots,i-1,i+1,\dots,v}(x)$$

Using this formula repeatedly, 得

x_0	$p_0(x)$				
x_1	$p_1(x)$	$p_{0,1}(x)$			
x_2	$p_2(x)$	$p_{1,2}(x)$	$p_{0,1,2}(x)$		
x_3	$p_3(x)$	$p_{2,3}(x)$	$p_{1,2,3}(x)$	$p_{0,1,2,3}(x)$	
x_4	$p_4(x)$	$p_{3,4}(x)$	$p_{2,3,4}(x)$	$p_{1,2,3,4}(x)$	$p_{0,1,2,3,4}(x)$

此處每一連續多項式可由前一行之兩個相鄰多項式所決定. 令

$$s_{ij} = p_{i-j,i-j+1,\dots,i-1,i}(x)$$

其中 s_{ij} 表由 $(j+1)$ 個結點 $x_{i-j}, x_{i-j+1}, \dots, x_{i-1}, x_i$, 次數爲 j 之插值多項式. 因此, we can rewrite the recurrence relation above as

$$\begin{aligned} s_{ij}(x) &= \left(\frac{x - x_{i-j}}{x_i - x_{i-j}}\right) s_{i,j-1}(x) + \left(\frac{x_i - x}{x_i - x_{i-j}}\right) s_{i-1,j-1}(x) \\ &= \frac{(x - x_{i-j}) s_{i,j-1}(x) - (x - x_i) S_{i-1,j-1}(x)}{x_i - x_{i-j}} \end{aligned}$$

x_0	$s_{00}(x)$				
x_1	$s_{10}(x)$	$s_{11}(x)$			
x_2	$s_{20}(x)$	$s_{21}(x)$	$s_{22}(x)$		
x_3	$s_{30}(x)$	$s_{31}(x)$	$s_{32}(x)$	$s_{33}(x)$	
x_4	$s_{40}(x)$	$s_{41}(x)$	$s_{42}(x)$	$s_{43}(x)$	$s_{44}(x)$

Change the notation by making the superscript to the degree of the polynomial for theoretical results.

Define constant polynomial

$$P_i^0(x) = y_i, \quad \text{for } \ 0 \leq i \leq n.$$

$$P_i^j(x) = \frac{x - x_{i-j}}{x_i - x_{i-j}} P_i^{j-1}(x) + (\frac{x_i - x}{x_i - x_{i-j}}) P_{i-j}^{j-1}(x)$$

其 中 $1 \leqq j \leqq n$ while $j \leqq i \leqq n$.

定理 **2. Interpolation properties**

The polynomial P_i^j defined above interpolate as follows.

$$P_i^j(x_k) = y_k, \quad 0 \leqq i - j \leqq k \leqq i \leqq n$$

證明 **1.** 用歸納法證明 *(省略)*.

Existence of Interpolating polynomial

The Lagrange interpolation formula proves the existence of an interpolating polynomial for any table of values. There is another constructive way of proving the fact, and it leads to a different formula (is called the **Newton form of Interpolating polynomial**). 此種方法也稱爲 Newton algorithm.

假設我們已成功找到 a polynomial p 滿足表格上部分點 (設爲 $k+1$ 個), 則

$$p(x_i) = y_i \quad \text{for} \quad 0 \leq i \leq k,$$

將此多項式表成 $p_k(x)$. 欲求一新多項式滿足前 $k+2$ 個點, 可記爲

$$p_{k+1}(x) = p_k(x) + c(x - x_0)(x - x_1) \cdots (x - x_k) \tag{4.5}$$

其中 c 是待定常數, 即只要求得常數 c 則 p_{k+1} 即得到.

由上式知 $p_{k+1}(x)$ 滿足 $x_0, x_1, ..., x_k$ 等 $k+1$ 個點, 即

$$p_{k+1}(x_j) = p_k(x_j) = y_j, \quad 0 \le j \le k.$$

現在考慮第 $k+2$ 個點, $(x_{k+1},\, y_{k+1})$. 我們假設 $p_{k+1}(x_{k+1}) = y_{k+1}$ 成立, 則由

$$p_{k+1}(x_{k+1}) = p_k(x_{k+1}) + c(x_{k+1} - x_0)(x_{k+1} - x_1)\cdots(x_{k+1} - x_k)$$

即可解得

$$c = \frac{y_{k+1} - p_k(x_{k+1})}{(x_{k+1} - x_0)(x_{k+1} - x_1)\cdots(x_{k+1} - x_k)} \tag{4.6}$$

Since $x_{k+1} - x_i \ne 0, \ \forall\, 0 \le i \le k$. 通常以 a_k 表此常數 c. 依此程序每增加一點即可求得一新多項式, 直到最後得到 p_n 通過所給的 $n+1$ 個點. 以上即是一個歸納推理(inductive reasoning)之例子.

定理 **3.** *(***Existence of Polynomial Interpolation Theorem***)*
若點 $x_0, x_1, ..., x_n$ 是相異的, 則對任意實數 $y_0, y_1, ..., y_n$, 存在唯一之多項式 p 其次數不大於 n, 使得

$$p(x_i) = y_i, \quad 0 \le i \le n.$$

★ 註: 此定理建立兩個部分:

1. 用歸納法推理: 開始時由0次多項式, 1次多項式, ..., 至 $n+1$ 個點時最多爲n次多項式.

2. 此多項式 p 是唯一之選擇—茲證明如下:

證明 **2.** 若有另一多項式 q 滿足

$$q(x_i) = y_i, \quad for \quad 0 \le i \le n$$

則 q 之次數最高爲 n. 考慮 $h(x) = p(x) - q(x)$, 其次數最高爲 n 且

$$h(x_i) = p(x_i) - q(x_i) = y_i - y_i = 0, \quad \forall\, 0 \le i \le n$$

Recall that *"A nonzero polynomial of degree n can have at most n roots"*. 因此 $h(x) \equiv 0$, 爲零多項式. 因此 $p(x) = q(x)$ 得證唯一性. $\sharp$

由前述之"歸納推理"所建立之 interpolating polynomial 稱為**the Newton form of the interpolating polynomial** (或稱為 **Newton algorithm**). 其可表成

$$p(x) = a_0 + \sum_{i=1}^{n} a_i \left[\prod_{j=0}^{i-1} (x - x_j) \right]. \tag{4.7}$$

其 "nested form" 為

$$\begin{aligned} p(x) &= a_0 + (x - x_0)(a_1 + (x - x_1)(a_2 + \cdots + (x - x_{n-1})a_n) \cdots) \\ &= (\cdots((a_n(x - x_{n-1}) + a_{n-1})(x - x_{n-2}) + a_{n-2}) \cdots)(x - x_0) + a_0. \end{aligned}$$

★ 我們之目的: 給一 t 值, 欲計算 $p(t)$ 之值—應用 Nested multiplication. 我們自然地由最內括弧 (the innermost parentheses) 開始, 故連續地形成下列之量:

$$\begin{aligned} v_0 &= a_n \\ v_1 &= v_0(t - x_{n-1}) + a_{n-1} \\ v_2 &= v_1(t - x_{n-2}) + a_{n-2} \\ &\cdots \\ v_n &= v_{n-1}(t - x_0) + a_0 \end{aligned}$$

則最後之 the quantity v_n 即是 $p(t)$, 故 pseudocode 可寫成

```
Procedure pt = Eval(a, x, t) %% evaluate $p(t)$
real array $\mathbf{a} = [a_i]_{0:n}$,    $\mathbf{x} = [x_i]_{0:n}$
$v = a_n$
for $i = n - 1$ to step $-1$ do
    $v = v(t - x_i) + a_i$
end for
end Procedure
```

其中 $[x_i]_{0:n}$ 為已知 nodes, 但 $[a_i]_{0:n}$ 是待定係數.

例題 **3. Using the Newton Algorithm to find the interpolating polynomial of least degree for this table:**

x	0	1	-1	2	-2
y	-5	-3	-15	39	-9

解答:

總共有 *5* 個點, 故我們將建立 *5* 個多項式. 表成 $p_0(x)$, $p_1(x)$, $p_2(x)$, $p_3(x)$ 和 $p_4(x)$.

1. 過點 $(0, -5)$ 之多項式為 $p_0(x) = -5$.

2. 設過點

x	0	1
y	-5	-3

之多項式為 $p_1(x)$. 則

$$\begin{aligned} p_1(x) &= p_0(x) + a_1(x - x_0) \\ &= -5 + a_1(x - 0). \end{aligned}$$

因 $p_1(1) = -3$, 故 $-5 + a_1 = -3 \;\Rightarrow\; a_1 = 2$

$\therefore p_1(x) = -5 + 2x$.

3. 設過點

x	0	1	-1
y	-5	-3	-15

之多項式為 $p_2(x)$, 則

$$\begin{aligned} p_2(x) &= p_1(x) + a_2(x - x_0)(x - x_1) \\ &= (-5 + 2x) + a_2\, x(x - 1) \end{aligned}$$

由 $p_2(-1) = -15$, 解得 $a_2 = -4$, 故得 $p_2(x) = -5+2x-4x(x-1)$.
繼續此法可求得

$$p_3(x) = -5 + 2x - 4x(x - 1) + 8x(x - 1)(x + 1)$$

$$p_4(x) = -5+2x-4x(x-1)+8x(x-1)(x+1)+3x(x-1)(x+1)(x-2)$$

即為所求. $\sharp$

註: 在 Maple 中可用指令
> interp$([0, 1, -1, 2, -2], [-5, -3, -15, 39, -9], x)$;
而求得 $p_4(x) = 3x^4 + 2x^3 - 7x^2 + 4x - 5$.

例題 **4.** 將上例中之 $p_4(x)$ 寫成 *nested form* 且計算 $p_4(3)$ 之值

4.2 Divided Differences(均差分)

Calculating Coefficients a_j using Divided Differences
假設所給 $n+1$ 個點 $x_0, x_1, ..., x_n$ 是相異的 (distinct) 且其函數值如下表:

x	x_0	x_1	...	x_n
$f(x)$	$f(x_0)$	$f(x_1)$	...	$f(x_n)$

則由前述之定理知: 必存在唯一之多項式 $p_n(x)$, 其次數最高爲 n, 且滿足

$$p_n(x_i) = f(x_i), \quad \text{for } \ i = 0, 1, 2, ..., n.$$

我們已證明 $p_n(x)$ 可表成 Newton form

$$\begin{aligned} p_n(x) &= a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \cdots + a_n(x-x_0)\cdots(x-x_{n-1}) \\ &= a_0 + \sum_{i=1}^{n} a_i \left(\prod_{j=0}^{i-1} (x-x_j) \right) \qquad (4.8) \\ & \qquad (4.9) \end{aligned}$$

或 the recursive form

$$p_n(x) = p_{n-1}(x) + a_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})$$

欲求係數 $a_0, a_1, ..., a_n$ 即令 $x = x_0, x_1, ..., x_n$ 代入上式 (4.8) 即得

$$\begin{aligned} f(x_0) &= a_0 \\ f(x_1) &= a_0 + a_1(x_1 - x_0) \\ f(x_2) &= a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \\ & \cdots \\ & \text{etc.} \end{aligned}$$

In general,

$$f(x_k) = \sum_{i=0}^{k} a_i \prod_{j=0}^{i-1} (x_k - x_j) \quad (0 \le k \le n) \qquad (4.10)$$

由這些方程式可解出係數

$$\begin{aligned}
a_0 &= f(x_0) = f[x_0] \longrightarrow \text{即 } a_0 \text{ 依據 } f(x_0) \\
a_1 &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f[x_0, x_1] \longrightarrow \text{即 } a_1 \text{ 依據 } f(x_0) \text{ 和 } f(x_1) \\
a_2 &= \frac{f(x_2) - a_0 - a_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} \\
&= \frac{f(x_2) - f[x_0] - f[x_0, x_1](x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} \\
&= \frac{f[x_1, x_2] - f[x_0, x_1]}{(x_2 - x_0)} = f[x_0, x_1, x_2], \longrightarrow \text{即 } a_2 \text{ 依據 } f(x_0), f(x_1), f(x_2) \\
&\cdots \\
&\text{etc.}
\end{aligned}$$

故 a_k 依據 $f(x_0), f(x_1), ..., f(x_k)$ 因此可將 a_k 表成

$$\begin{aligned}
a_k &= f[x_0, x_1, ..., x_k] \\
&= \frac{f[x_1, x_2, ..., x_k] - f[x_0, x_1, ..., x_{k-1}]}{x_k - x_0}
\end{aligned} \tag{4.11}$$

這個值 $f[x_0, x_1, ..., x_k]$ 稱爲f 的 k 階均差分 (the divided difference of order k for f). 這些係數 $a_0, a_1, ..., a_n$ 由 (4.11) 所唯一決定.

例題 **5.** *For the table*

x	1	−4	0
$f(x)$	3	13	−23

determine the quantities $f[x_0], f[x_0,\ x_1],$ *and* $f[x_0,\ x_1,\ x_2]$.

解得 $a_0 = 3$, 故 $f[1] = 3$. $a_1 = -2$, 故 $f[1, -4] = -2$. $a_2 = 7$, 故 $f[1, -4, 0] = 7$.

Newton algorithm for computing the coefficients a_k:

$$a_k = \frac{y_k - p_{k-1}(x_k)}{(x_k - x_0)(x_k - x_1)...(x_k - x_{k-1})}$$

Procedure

```
$a_0 = y_0$
for $k = 1$ to $n$ do
    $d = x_k - x_{k-1}$
    $u = a_{k-1}$
    for $i = k-2$ to $0$ step $-1$ do
        $u = u(x_k - x_i) + a_i$        % 算 $p_{k-1}(x_k)$
        $d = d(x_k - x_i)$              % 分母
    end
    $a_k = (y_k - u)/d$
end
end procedure
```

以 f 的 k 階均差分來表示 **the Newton interpolating polynomial**

$$p_n(x) = \sum_{i=0}^{n} \left\{ f[x_0, x_1, ..., x_i] \prod_{j=0}^{i-1} (x - x_j) \right\} \tag{4.12}$$

其中規定 $\sum_{j=0}^{-1}(x - x_j) = 1.$

★ 注意到在 $p_n(x)$ 中, x^n 之係數爲 $f[x_0, x_1, ..., x_n]$, 因 x^n 項只發生在 $\prod_{j=0}^{n-1}(x - x_j)$. 故若 f 是一個 $\underline{\text{degree} \le n-1}$ 之 polynomial, 則 $f[x_0, x_1, ..., x_n] = 0$.

欲求 Coefficients a_k, $0 \le k \le n$, 由公式 (4.8) 知, 此計算可被performed recursively. 若已知 $a_0, a_1, ..., a_{k-1}$, 則

$$f(x_k) = a_k \prod_{j=0}^{k-1}(x_k - x_j) + \sum_{i=0}^{k-1} a_i \prod_{j=0}^{i-1}(x_k - x_j)$$

和

$$a_k = \frac{f(x_k) - \sum_{i=0}^{k-1} a_i \prod_{j=0}^{i-1}(x_k - x_j)}{\prod_{j=0}^{k-1}(x_k - x_j)}$$

即

$$f[x_0,x_1,...,x_k] = \frac{f(x_k) - \sum_{i=0}^{k-1} f[x_0,x_1,...,x_i]\prod_{j=0}^{i-1}(x_k-x_j)}{\prod_{j=0}^{k-1}(x_k-x_j)} \tag{4.13}$$

因此, 求 Coefficients 之 algorithm:

1. Set $f[x_0] = f(x_0)$.
2. For $k = 1,2,3...,n$, 計算上式 $f[x_0,x_1,...,x_k]$ 即得.

例題 **6.**

$$\begin{aligned} f[x_0] &= f(x_0) \\ f[x_0,x_1] &= \frac{f(x_1)-f(x_0)}{x_1-x_0} \\ f[x_0,x_1,x_2] &= \frac{f(x_2)-f[x_0]-f[x_0,x_1](x_2-x_0)}{(x_2-x_0)(x_2-x_1)} \\ f[x_0,x_1,x_2,x_3] & \end{aligned}$$

$$= \frac{f(x_3)-f[x_0]-f[x_0,x_1](x_3-x_0)-f[x_0,x_1,x_2](x_3-x_0)(x_3-x_1)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$$

★ 上面所列求 Coefficients 之 algorithm 易於 coding. 計算 $f[x_0]$, $f[x_0,x_1]$,..., $f[x_0,x_1,...,x_n]$ 共 Cost

- $\frac{1}{2}n(3n+1)$ additions / subtractions.
- $(n-1)(n-2)$ multiplications.
- n divisions excluding arithmetics on the indices.

★ 更好的 algorithm依據下面定理, 只用 $\frac{1}{2}n(n+1)$ divisions and $n(n+1)$ additions.

定理 **4. Recursive Property of Divided Differences**

The divided difference obey the formula

$$f[x_0,x_1,...,x_k] = \frac{f[x_1,x_2,...,x_k]-f[x_0,x_1,...,x_{k-1}]}{x_k-x_0}. \tag{4.14}$$

證明 **3.** 因 $a_k = f[x_0, x_1, ..., x_k]$ 爲插值多項式之 Newton form $p_k(x)$ 之係數, 故可說 $f[x_0, x_1, ..., x_k]$ 是 $p_k(x)$ *(degree* $\leq k$*)* 中 x^k 之係數, 因 $p_k(x)$ interpolates f *at* $x_0, x_1, ..., x_k$. 同理, $f[x_1, x_2, ..., x_k]$ 是多項式 $q(x)$ *(degree* $\leq k-1$*)* 中 x^{k-1} 之係數, 因 $q(x)$ interpolates f 在點 $x_1, x_2, ..., x_k$. 相同地, $f[x_0, x_1, ..., x_{k-1}]$ 是多項式 $p_{k-1}(x)$ *(degree* $\leq k-1$*)* 中 x^{k-1} 之係數, 因它是 p_{k-1} interpolates f 在點 $x_0, x_1, ..., x_{k-1}$. 這三個多項式 p_k, q, 和 p_{k-1} 有密切的相關:

$$p_k(x) = q(x) + \frac{x - x_k}{x_k - x_0}[q(x) - p_{k-1}(x)] \tag{4.15}$$

上式代入 x_0 得 $f(x_0)$:

$$\begin{aligned} q(x_0) + \frac{x_0 - x_k}{x_k - x_0}[q(x_0) - p_{k-1}(x_0)] &= q(x_0) - [q(x_0) - p_{k-1}(x_0)] \\ &= p_{k-1}(x_0) = f(x_0). \end{aligned}$$

Evaluating it at x_i, for $1 \leq i \leq k-1$, results in $f(x_i)$:

$$\begin{aligned} q(x_i) + \frac{x_i - x_k}{x_k - x_0}[q(x_i) - p_{k-1}(x_i)] &= f(x_i) + \frac{x_i - x_k}{x_k - x_0}[f(x_i) - f(x_i)] \\ &= f(x_i) \end{aligned}$$

Similarly, at x_k, we get $f(x_k)$:

$$q(x_k) + \frac{x_k - x_k}{x_k - x_0}[q(x_k) - p_{k-1}(x_k)] = q(x_k) = f(x_k)$$

由插值多項式之唯一性, *(4.15)* 式之右邊必爲 $p_k(x)$, 因此 *(4.15)* 成立. 最後比較 *(4.15)* 式兩邊之 x^k 係數, 即得到 *(4.14)* 式. *Indeed,* $f[x_1, x_2, ..., x_k]$
是 q 中 x^{k-1} 之係數 且 $f[x_0, x_1, ..., x_{k-1}]$
是 p_{k-1} 中 x^{k-1} 之係數 ♯

定理 **5.** ***Invariance Theorem***

The divided difference $f[x_0, x_1, ..., x_k]$ *is invariant under all permulations of the arguments* $x_0, x_1, ..., x_k$.

例如 $f[x_0,x_1,x_2]=f[x_1,x_0,x_2]$ 或 $f[z_0,z_1,...,z_k]=f[x_0,x_1,x_2,...,x_k]$, 其中 $(z_0,z_1,...,z_k)$ 是 $x_0,x_1,...,x_k$ 之一重排. 依此定理得 the recursive formula

$$\begin{cases} f[x_i]=f(x_i) \\ f[x_i,x_{i+1},...,x_{j-1},x_j]=\frac{f[x_{i+1},x_{i+2},...,x_j]-f[x_i,x_{i+1}...,x_{j-1}]}{x_j-x_i} \longrightarrow (13) \\ \qquad\qquad 0\le i\le j\le n. \end{cases}$$

由此公式即可建立一”均差分表” for a function f.

x	$f[\ ]$	$f[\ ,\ ]$	$f[\ ,\ ,\ ]$	$f[\ ,\ ,\ ,\ ]$
x_0	$f[x_0]$			
		$f[x_0,x_1]$		
x_1	$f[x_1]$		$f[x_0,x_1,x_2]$	
		$f[x_1,x_2]$		$f[x_0,x_1,x_2,x_3]$
x_2	$f[x_2]$		$f[x_1,x_2,x_3]$	
		$f[x_2,x_3]$		
x_3	$f[x_3]$			

例題 **7.** Construct a divided-difference diagram for the function f given in the following table, and write out the Newton form of the interpolating polynomial.

x	1	2/3	0	2
$f(x)$	3	13/4	3	5/3

x	$f[\ ]$	$f[\ ,\ ]$	$f[\ ,\ ,\ ]$	$f[\ ,\ ,\ ,\ ]$
1	3			
		$1/2$		
$3/2$	$13/4$		$1/3$	
		$1/6$		-2
0	3		$-5/3$	
		$-2/3$		
2	$5/3$			

故有,

$$\begin{aligned} p_3(x) &= 3+1/2(x-1)+1/3(x-1)(x-3/2)-2(x-1)(x-3/2)(x) \\ &= \underline{-2x^3+\frac{16}{3}x^2-\frac{10}{3}x+3}\sharp \end{aligned}$$

Algorithms and Pseudocode

假設 f 在 $x_0, x_1, ..., x_n$ 之值給定如 table，欲求均差分 (存放於矩陣 a_{ij} 中)

$$\begin{aligned} a_{ij} &\equiv f[x_i, x_{i+1}, ..., x_{i+j}] \\ &= \frac{f[x_{i+1}, x_{i+2}, ..., x_{i+j}] - f[x_i, x_{i+1}, ..., x_{i+j-1}]}{x_{i+j}-x_i}. \end{aligned}$$

```
Procedure Coef(I)
real array $[a_{ij}]_{(n+1)\times(n+1)}$, $[x_i]_{1:n+1}$
integer $i$, $j$, $k$
for $i = 1$ to $n+1$ do
    $a_{i0} = f(x_i)$
end for
for $j = 2$ to  $n$ do
    for $i = 1$ to  $n-j+1$ do
    $a_{ij} = (a_{i+1,j-1} - a_{i,j-1})/(x_{i+j-1} - x_i)$
   end for
end for
```

其計算結果存在矩陣 $[a_{ij}]_{(n+1)\times(n+1)}$, 其中第一列即是所要之係數 $a_0, a_1, ..., a_n$.

Algorithms 可分爲兩個 procedures:

- 第一個稱爲 "coef": 計算係數 $a_0, a_1, ..., a_n$.
- 第二個稱爲 "eval" $\longrightarrow$ 改爲 Intp_poly 計算 the interpolating polynomial, 在$\underline{t \text{ 的函數值}}$.

註: 在計算 the divided differences 時只用來求插值多項式之 Newton form , 故只需要 store 係數 $a_0, a_1, ..., a_n$, 即只要用一個$\underline{\text{向量}}$即可儲存 $[a_0, a_1, ..., a_n]$. 因此 procedure 可改爲

```
Procedure Coef(II)
for i = 0 to n do
   $a_i = f(x_i)$
end
for j = 1 to n do
   for i = n to j step -1 do
      $a_i = \frac{a_i - a_{i-1}}{x_i - x_{i-j}}$
   end
end
```

例題 **8.** Write a pseudocode that determines the Newton form of the interpolating polynomial p for $\sin x$ at ten equidistant points in $[0, 1.6875]$. The code should print the value of $\sin x - p(x)$ at 37 equally spaced points in the same interval.

解答:

要取10個點故需分成 9 個子區間, 其長度爲 $h = 0.1875$. 10個點爲 $x_i = ih$ for $i = 0, 1, 2, ..., 9$. 得到多項式之後要計算 $\sin(x) - p$ 在37點之値. 故令 $t_j = jh/4$ for $i = 0, 1, ..., 36$. 程式如下:

```
Program $Main$
integer parameter $n \longleftarrow 9$
$h = 1.6875/n$
for $k = 0$ to $n$ do
    $x_k = kh$
    $y_k = \sin(x_k)$
end for
call $Coef(n, (x_i), (y_i))$
output $(a_i)$
for j $= 1$ to $4n$ do
    $t = jh/4$
    $p = Eval(n, (x_i), (a_i), t)$      % call $Eval.m$
    $err = |sin(t) - p|$      % compute the errors
end for
end program $Main$
```

主程式 (pseudocode)

在應用 MATLAB 之內建程式, 本程式也可寫成如下: 中等距之點可用

```
$> a = 0,\ b = 1,\ n = 9$
$> X = linespace(a, b, n+1)$
$> Y = \sin(X);$
$> p = polyfit(X, Y, n);$
$> Z = linespace(a, b, 4*n+1)$
$> F = polyval(p, Z)$
$> E = \sin(Z) - F$
$> plot(X, Y, 'o', Z, F, '-')$
```

$$\begin{aligned}
p_{0,1}(x) &= \frac{(t-x_1)p_0(x)-(t-x_0)p_1(x)}{x_0-x_1} \\
p_{1,2}(x) &= \frac{(t-x_2)p_1(x)-(t-x_1)p_2(x)}{x_1-x_2}
\end{aligned}$$

4.3 Errors in polynomial interpolation

當一函數 f 在區間 $[a, b]$ 被一插值多項式 $p(x)$ 所近似, 則 $f(x_j)-p(x_j)=0$ 對所有所給的節點 x_j. 一個很自然的期望是對於區間 $[a, b]$ 上任一點 x, $p(x)$ 是 $f(x)$ 很好的近似值.

問題一: 當所給的節點愈多, 則近似的一致性會愈好?
答案是: 否定的對於沒有好行爲之函數. 例如一些特殊不連續函數 (稱爲 Pathological examples). 比如: **The Dirichlet function δ:**

$$\delta(x)=\begin{cases}1, & x\in Q',\\ 0, & x\in Q.\end{cases}$$

若選擇有理數當節點, 則所得之 $p(x)\equiv 0$ 且 $f(x)-p(x)=0,\ \forall\ x\in Q$ 但 $f(x)-p(x)=1,\ \forall\ x\in Q'$.

問題二: 若 f 是有好行爲之函數, 則插入之 nodes 愈多 (即 p_n 之次數愈高時), $\max_{x\in[a,\,b]}|f(x)-p_n(x)|$ 會愈來愈小嗎? 答案是否定的對於某些函數等距節點之選擇, 所得之插值多項式有 'n愈大誤差愈大'之現象. 例如: **The Runge function**

$$f(x)=\frac{1}{1+x^2},\quad x\in[-5,\,5].$$

設有一插值多項式 $p_n(x)$ 插入此函數在區間 $[-5, 5]$ 上之 $n+1$ 個等距點 (equally spaced points)(包含端點), 則 $\max_{-5\leqq x\leqq 5}|f(x)-p_n(x)|$ 並不會隨 n 之增大而愈來愈小. 而且有

$$\lim_{n\to\infty}\max_{-5\leqq x\leqq 5}|f(x)-p_n(x)|=\infty$$

之現象. 因此, 次數愈高的插值多項式 $p_n(x)$ 並不是愈好. 其問題出在等距節點之選擇. 有一種較好節點之選擇是Chebychev nodes. 其在區間 $[-1, 1]$ 爲:

$$x_i=\cos\left(\frac{i}{n}\pi\right),\quad 0\leqq i\leqq n.$$

其對應於任意區間 $[a, b]$ 可由線性映射 (linear mapping)得到

$$x_i=\frac{1}{2}(a+b)+\frac{1}{2}(b-a)\cos\left(\frac{i}{n}\pi\right),\quad 0\leqq i\leqq n.$$

注意到這些點的足標是由右至左.

定理 **6. Interpolation Errors Theorem I**

若 *degree*$\leqq n$ 之多項式 p 插 f 於 $n+1$ 個相異點 $x_0, x_1, ..., x_n$, 其中 $x_i \in [a, b], \forall 0 \leqq i \leqq n$. 且若 $f^{(n+1)}$ 是連續, 則 $\forall x \in [a, b]$, $\exists \xi \in (a, b)$ 使得

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=1}^{n} (x - x_i) \tag{4.16}$$

證明 **4.** 省略.

定理 **7. Interpolation Errors Theorem II**

若函數 f 有 $f^{(n+1)}$ 在 $[a, b]$ 連續 且 $|f^{(n+1)}(x)| \leqq M, \forall x \in [a, b]$, *Let p be the polynomial of degree $\leqq n$ that interpolates f at $n+1$ equally spaced nodes in* $[a, b]$*(包含端點)*, 則 *on* $[a, b]$

$$|f(x) - p(x)| \leqq \frac{M}{4(n+1)} \left(\frac{b-1}{n} \right)^{n+1}. \tag{4.17}$$

定理 **8. Interpolation Errors Theorem III**

若 p 是一個次數爲 n 之多項式, 插函數 f 於節點 $x_0, x_1, ..., x_n$, 則對任意 x 不是節點, 有

$$f(x) - p(x) = f[x_0, x_1, ..., x_n, x] \prod_{i=0}^{n} (x - x_i). \tag{4.18}$$

證明 **5.** 省略.

定理 **9. Divided Differences & Derivatives Theorem**

若 $f^{(n)}$ 在 $[a, b]$ 連續且 $x_0, x_1, ..., x_n$ 是 $[a, b]$ 上 $n+1$ 個相異點 則存在一數 $\xi \in (a, b)$ 使得

$$f[x_0, x_1, ..., x_n] = \frac{1}{n!} f^{(n)}(\xi). \tag{4.19}$$

證明 **6.** 省略.

定理 **10. Divided Differences Corollary**

若 f 是一個次數爲 n 之多項式, 則當 $i \geqq n+1$ 時, 所有均差分 $f[x_0, x_1, ..., x_i]$ 都爲零. *(see Example 2, p. 168)*

4.4 Hermite Interpolation

Porblem: Given a set of $n+1$ distinct data with x_i, $f(x_i)$, and $f'(x_i)$, we want to find a polynomial agrees with f and f' at the points.

Osculating polynomials generalize both the Taylor polynomials and Lagrange polynomials.

定義 **1.** *Let $x_0, x_1, ..., x_n$ be $n+1$ distinct numbers in $[a,b]$ and m_i be nonnegative integers associated with x_i for $i = 1, 2, \ldots, n$. Suppose that $f \in C^m[a,b]$ and $m = \max_{0 \le i \le n} m_i$. The* **osculating polynomial** *approximating f is the polynomial $P(x)$ of least degree such that*

$$\frac{d^k P(x_i)}{dx^k} = \frac{d^k f(x_i)}{dx^k} \quad \textit{for each } \; i = 0, 1, \ldots, n \; \textit{ and } \; k = 0, 1, \ldots m_i.$$

★ The osculating polynomial is the polynomial of least degree with the property that it agrees with f and all its derivatives of order $\le m_i$ at x_i for each i=0,1,...,n. The degree of this osculating polynomial is at most

$$M = \sum_{i=0}^{n} m_i + n$$

since the number of conditions to be satisfied is $\sum_{i=0}^{n}(m_i + 1) = \sum_{i=0}^{n} m_i + (n+1)$, and a polynomial of degree M has $M+1$ coefficients that can be used to satisify these conditions.

Notes:

1. The case when $n = 0$, the osculating polynomial approximating f is simply the m_0th Taylor polynomial for f at x_0.
2. The case when $m_i = 0$ for each i, the osculating polynomial is the nth Lagrange polynomial interpolating f on $x_0, x_1, \ldots, x_n$.
3. The case when $m_i = 1$ for each $i = 0, 1, \ldots, n$ gives a class called the Hermite polynomials.

定理 **11. Hermite Polynomial**

If $f \in C^1[a,b]$ and $x_0, x_1, \ldots, x_n \in [a,b]$ are distinct, the unique polynomial of least degree agreeing with $f(x_i)$, and $f'(x_i)$ at $x_0, x_1, \ldots, x_n$ is the Hermite polynomial of degree at most $2n+1$ given by

$$H_{2n+1}(x) = \sum_{j=0}^{n} f(x_j) H_{n,j}(x) + \sum_{j=0}^{n} f'(x_j) \hat{H}_{n,j}(x),$$

where

$$H_{n,j}(x) = [1 - 2(x - x_j) L'_{n,j}(x_j)] L^2_{n,j}(x)$$

and

$$\hat{H}_{n,j}(x) = (x - x_j) L^2_{n,j}(x).$$

In this context, $L_{n,j}(x)$ denotes the jth Lagrange coefficient polynomial of degree n (also called the cardinal polynomials).

PROOF:

例題 **9.** *Use the Hermite polynomial tha agrees with the data, listed in the following table, to find an approximation of $f(1.5)$.*

k	x_k	$f(x_k)$	$f'(x_k)$
0	1.3	0.6200860	-0.5220232
1	1.6	0.4554022	-0.5698959
2	1.9	0.2818186	-0.5811571

Solution:

定理 **12. Hermite polynomial Error Formula**

If $f \in C^{2n+2}[a,b]$, then

$$f(x) = H_{2n+1}(x) + \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!}(x-x_0)^2 \cdots (x-x_n)^2$$

for some $\xi(x)$ (which is unknown) $\in (a,\, b)$.

Notes:

To find the Hermite polynomials by the Theorem that the need to determine and evaluate the Lagrange polynomials and their derivatives makes the procedure *tedious* even for small value of n. An alternative method is based on the *Newton interpolating divided-difference formula* for the Lagrange polynomial at $x_0, x_1, \ldots, x_n$,

$$P_n(x) = f[x_0] + \sum_{k=0}^{n} f[x_0, x_1, \ldots, x_k](x-x_0)\cdots(x-x_{k-1}),$$

and the connection between the nth devided difference and the nth derivative of f, as outlined in Theorem 9 in the previous section.

定理 **13. Divided-Difference Form of Hermite polynomial**

If $f \in C^1[a,b]$ and $x_0, x_1, \ldots, x_n \in [a,b]$ are distinct, then

$$H_{2n+1}(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, z_1, \ldots, z_k](x-z_0)(x-z_1)\cdots(x-z_{k-1}),$$

where $z_{2k} = z_{2k+1} = x_k$ *and* $f[z_{2k}, z_{2k+1}] = f'(x_k)$ *for each* $k = 0, 1, \ldots, n.$

z	$f(z)$	$f[\quad,\quad]$	$f[\quad,\quad,\quad]$
$z_0 = x_0$	$f[z_0] = f(x_0)$		
		$f[z_0, z_1] = f'(x_0)$	
$z_1 = x_0$	$f[z_1] = f(x_0)$		$f[z_0, z_1, z_2] = \frac{f[z_1,z_2]-f[z_0,z_1]}{z_2-z_0}$
		$f[z_1, z_2] = \frac{f[z_2]-f[z_1]}{z_2-z_1}$	
$z_2 = x_1$	$f[z_2] = f(x_1)$		$f[z_1, z_2, z_3] = \frac{f[z_2,z_3]-f[z_1,z_2]}{z_3-z_1}$
		$f[z_2, z_3] = f'(x_1)$	
$z_3 = x_1$	$f[z_3] = f(x_1)$		$f[z_2, z_3, z_4] = \frac{f[z_3,z_4]-f[z_2,z_3]}{z_4-z_2}$
		$f[z_3, z_4] = \frac{f[z_4]-f[z_3]}{z_4-z_3}$	
$z_4 = x_2$	$f[z_4] = f(x_2)$		$f[z_3, z_4, z_5] = \frac{f[z_4,z_5]-f[z_3,z_4]}{z_5-z_3}$
		$f[z_4, z_5] = f'(x_2)$	
$z_5 = x_2$	$f[z_5] = f(x_2)$		

例題 **10.** *As Example 1, to find the Hermite polynomial agrees with the given data.*

SOLUTION:

4.5 數値微分和理查德森外推法 (Numerical Differentiation and Richardson Extrapolation)

First-Derivative formulas via Taylor Series

Recall the definition of $f'(x)$:

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

We have

$$f'(x) \approx \frac{1}{h}[f(x+h) - f(x)] \tag{4.20}$$

What error is involved in this formula? Use Taylor's Theorem from Section 1.2:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(\xi)$$

其中 ξ 介於 x 和 $x+h$ 間. Rearranging this equation gives

$$f'(x) = \frac{1}{h}[f(x+h) - f(x)] - \frac{1}{2}hf''(\xi). \tag{4.21}$$

由此公式知, 公式 (4.20) 成立且 The error term $= -\frac{1}{2}hf''(\xi) = O(h)$. 故有 error $\longrightarrow 0$ as $h \longrightarrow 0$. 此種誤差又稱爲截尾誤差 (the truncation error).

Consider now

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2!}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + \cdots \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2!}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + \cdots \end{aligned} \tag{4.22}$$

兩式相減再除以 $2h$ 得:

$$f'(x) = \frac{1}{2h}[f(x+h) - f(x-h)] - \frac{h^2}{3!}f'''(x) - \frac{h^4}{5!}f^{(5)}(x) - \cdots. \tag{4.23}$$

故得更好之近似 $f'(x)$ 之公式:

$$f'(x) \approx \frac{1}{2h}[f(x+h) - f(x-h)] \tag{4.24}$$

with the error term is $-\frac{1}{6}h^2 f'''(\xi) = O(h^2)$.

Richardson Extrapolation

假設函數 f 和 x 固定, 定義一 h 之函數

$$\varphi(h) = \frac{1}{2h}[f(x+h) - f(x+h)] \tag{4.25}$$

其中 $\varphi(h)$ 近似 $f'(x)$ 誤差爲 $O(h^2)$. 我們的目的是計算 $\lim_{h\to 0}\varphi(h)$ 因

$$\lim_{h\to 0}\varphi(h) = f'(x).$$

若取 $h = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$, 則 $\varphi(h) \longrightarrow f'(x)$.

將 (4.23) 改寫爲

$$\begin{aligned} f'(x) &= \frac{1}{2h}[f(x+h) - f(x-h)] + a_2h^2 + a_4h^4 + a_6h^6 + \cdots \\ &= \varphi(h) + a_2h^2 + a_4h^4 + a_6h^6 + \cdots \end{aligned} \tag{4.26}$$

因此得

$$\varphi(h) = f'(x) - a_2h^2 - a_4h^4 - a_6h^6 - \cdots . \tag{4.27}$$

今令 $h/2$ 取代 h 得

$$\varphi\left(\frac{h}{2}\right) = f'(x) - a_2\left(\frac{h}{2}\right)^2 - a_4\left(\frac{h}{2}\right)^4 - \cdots$$

將 Eq.(4.27) 減去上式乘 4 得

$$\varphi(h) - 4\varphi\left(\frac{h}{2}\right) = -3f'(x) - \frac{3}{4}a_4h^4 - \frac{15}{16}a_6h^6 - \cdots$$

再除以 -3 得

$$\frac{4}{3}\varphi\left(\frac{h}{2}\right) - \frac{1}{3}\varphi(h) = f'(x) + \frac{1}{4}a_4h^4 + \frac{5}{16}a_6h^6 + \cdots . \tag{4.28}$$

將上式右邊定義爲 $\varphi_1(h)$, 而原來 Eq.(4.25) 之 φ 表爲 $\varphi_0(h)$. 即

$$\frac{4}{3}\varphi_0\left(\frac{h}{2}\right) - \frac{1}{3}\varphi_0(h) := \varphi_1(h) \approx f'(x).$$

因此有 $\varphi_1(h) \approx f'(x)$ with error $O(h^4)$. 同理, 將 $\varphi_1(h)$ 減去 $16\times\varphi_1(h/2)$ 再除以 -15 可得

$$\frac{16}{15}\varphi_1\left(\frac{h}{2}\right) - \frac{1}{15}\varphi_1(h) := \varphi_2(h) \approx f'(x)$$

with error $O(h^6)$. Repeat the same procedure, 我們可得

$$\varphi_m(h) = \frac{4^m \cdot \varphi_{m-1}(h/2) - \varphi_{m-1}(h)}{4^m - 1} \approx f'(x)$$

with error $O(h^{2(m+1)})$, for $m = 1, 2, 3, \ldots$. 更一般性地, 我們有

$$\varphi_m(h/2^n) = \frac{4^m \cdot \varphi_{m-1}(h/2^n) - \varphi_{m-1}(h/2^{n-1})}{4^m - 1}, \quad 1 \leq m \leq n. \tag{4.29}$$

在第五章中 Romberg's algorithm 的導出有相同之情況. 因此我們將給此種方法更一般化的討論. 設函數 φ 有性質如下:

$$\varphi(h) = L - \sum_{k=1}^{\infty} a_{2k} h^{2k} \tag{4.30}$$

其中 a_{2k} 是未知係數, L 是欲近似之值. 假定 $\varphi(h)$ 可被計算對於任意 $h > 0$. 今選擇一個方便的 h, 我們的目的是要精確地近似 L, 藉著計算下列各數:

$$\begin{aligned} D(n,0) &= \varphi\left(\frac{h}{2^n}\right), \quad (n \geq 0) \\ &= L + \sum_{k=1}^{\infty} A(k,0)\left(\frac{h}{2^n}\right)^{2k} \end{aligned} \tag{4.31}$$

其中 $A(k,0) = -a_{2k}$ 且 $D(n,0) \approx L$, since $\lim_{x\to 0}\varphi(x) = L$.

Richardson extrapolation (recurrence) formula

$$D(n,m) = \frac{4^m}{4^m-1}D(n,m-1) - \frac{1}{4^m-1}D(n-1,m-1), \quad (1 \leq m \leq n). \tag{4.32}$$

定理 14. Richardson Extrapolation Theorem
The quantities defined in the Richardson Extrapolation process (4.32) obey the equation

$$D(n,\, m) = L + \sum_{k=m+1}^{\infty} A(k,\, m) \left(\frac{h}{2^n}\right)^{2k} \quad (0 \le m \le n) \qquad (4.33)$$

where $A(k,m) = A(k,m-1)\left(\frac{4^m - 4^k}{4^m - 1}\right), \quad 0 \le m \le k.$

證明 7. 應用 *inductive proof:* 依據 *hypothesis*(假設), 若 $m = 0$, 則 *Eq.(4.32)* 成立. 故我們可以假定 *(assume), Eq.(4.32)* 式成立對每一個 $m-1$. 欲證對於每一個 m *Eq.(4.32)* 也成立:
由 *Eq.(4.33)* 代入 *Eq.(4.32)* 式中, *for fixed m,* 則有

$$\begin{aligned} D(n,\, m) &= \frac{4^m}{4^m - 1}\left[L + \sum_{k=m}^{\infty} A(k, m-1)\left(\frac{h}{2^n}\right)^{2k}\right] \\ &\quad - \frac{1}{4^m - 1}\left[L + \sum_{k=m}^{\infty} A(k, m-1)\left(\frac{h}{2^{n-1}}\right)^{2k}\right] \\ &= L + \sum_{k=m}^{\infty} A(k, m-1)\left(\frac{4^m - 4^k}{4^m - 1}\right)\left(\frac{h}{2^n}\right)^{2k} \end{aligned}$$

經化簡後上式可改寫成

$$D(n,m) = L + \sum_{k=m+1}^{\infty} A(k,m)\left(\frac{h}{2^n}\right)^{2k}$$

注意到 $A(m,\, m) = 0$.
故 *(4.33)* 式成立 *for all m, and the induction is complete* $\sharp$.

★ 在 Eq.(4.33) 中, $D(n,m) \longrightarrow L$ very rapidly, since 餘式項之第一項爲 $(h/2^n)^{2m+2}$.

★ 在應用上, 可建立一 triangular array:

$$\begin{array}{|c|c|c|c|c|}
D(0,0) & & & & \\
D(1,0) & D(1,1) & & & \\
D(2,0) & D(2,1) & D(2,2) & & \\
\cdots & \cdots & \cdots & \cdots\cdots & \\
D(N,0) & D(N,1) & D(N,2) & \cdots\cdots & D(N,N)
\end{array}$$

Algorithm of Richardson extrapolation

1. Write a procedure for φ.

2. Decide on suitable values for N and h.

3. For $i = 0, 1, ..., n$, compute $D(i, 0) = \varphi(h/2^i)$.

4. For $0 \leq i \leq j \leq N$, compute

$$D(i,\, j) = D(i,\, j-1) + (4^j - 1)^{-1}[D(i,\, j-1) - D(i-1,\, j-1).$$

注意: 此種寫法在計算上可減少誤差 (improve its numerical properties).

例題 **11.** *Write a procedure to compute the derivative of a function at a point by using Eq.(4.24) and Richardson extrapolation.*
解答 :
Input a function f, a specific point x, a value of h, and a number n signifying(指定) 多少 rows in 在上式中 to be computed.

```
procedure DerivativeRE(f, x, n, h), array $d_{ij}$
**for** $i = 1$ **to** (n+1)
   $d_{i1} = [f(x+h) - f(x-h)]/(2h)$
   **for** j $= 1$ **to** $i$
      $d_{i,j+1} = d_{ij} + (d_{ij} - d_{i-1,j})/(4^j - 1)$
   **end**
   $h = h/2$
**end**
**end** Procedure
```

To test the procedure, choose $f(x) = \sin x$ with $x_0 = 1.2309594154$ and $h = 1$ then

$$f'(x) = \cos x \quad \text{and} \quad f'(x_0) = \frac{1}{3}$$

```
program Main
$n = 10;$    $h = 1;$
$x = 1.2309594154;$
Call  DerivativeRE $(f, x, n, h)$
output $(d_{ij})$
end program Main
```

First-Derivative Formulas via Interpolation polynomials

Assume that a function f is approximated by a polynomial p so that $f(x) \approx p(x)$, then $f'(x) \approx p'(x)$ as a consequence. Of caurse, this strategy should be used very cautiously because the behavior of the interpolating polynomial can be oscillatory.

In pratice, the approximating polynomial p is often determined by interpolation at a few points. For example, suppose that p is the polynomial of *degree* ≤ 1, that interpolates f at two nodes, x_0 and x_1, then from eq.(8) in Sec. 4.1 with $n = 1$

$$p_1(x) = f(x_0) + f[x_0, x_1](x - x_0)$$

故

$$f'(x) \approx p_1'(x) = f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{4.34}$$

若 $x_0 = x$ and $x_1 = x + h$, 則上式即是 the Eq.(4.20)

$$f'(x) = \frac{1}{h}[f(x+h) - f(x)]$$

若 $x_0 = x - h$, and $x_1 = x + h$ 則 Eq.(4.34) becomes the Eq.(4.24)

$$f'(x) = \frac{1}{2h}[f(x+h) - f(x-h)]$$

Now consider interpolation with three nodes x_0, x_1, and x_2, The interpolating polynomial is obtained from eq.(8) in Sec.4.1

$$p_2(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

and its derivative is

$$p_2'(x) = f[x_0, x_1] + f[x_0, x_1, x_2](2x - x_0 - x_1) \tag{4.35}$$

The right-hand side 包含兩項: the first term is Eq.(4.34) + the second term is a refinement or correction term.
若 Eq.(4.35) 用來近似 $f'(x)$ 當 $x = \frac{1}{2}(x_0 + x_1)$, 則 Eq.(4.35) 中之 correction term = 0. 因此, Eq.(4.35) 比其他情況 more accurate than the case of interpolating at two nodes (Eq.(20)).

★ 如一般之 error analysis:
Suppose p_n is the polynomial of degree $\leq n$ that interpolates f at the nodes $x_0, x_1, ..., x_n$, 則依據 first (error) theorem in sec.4.2

$$f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) w(x)$$

其中 ξ depends on x and $w(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$. 兩邊微分得

$$f'(x) - p'(x) = \frac{1}{(n+1)!} w(x) \frac{d}{dx} f^{(n+1)}(\xi) + \frac{f^{(n+1)}(\xi)}{(n+1)!} w'(x) \tag{4.36}$$

此處必須 assume that $f^{(n+1)}(\xi)$ 是 x 的可微分函數, 若 $f^{(n+2)}$ 存在且連續, 則它是事實. 注意到下列各項觀察:

1. The first observation: In Eq.(4.36) $w(x) = 0$ at each node x_i, $0 \leq i \leq n$. 若欲計算 $f'(x_i)$ 之値, 則 the resulting equation is simpler:

$$f'(x_i) = p_n'(x_i) + \frac{1}{(n+1)!} f^{(n+1)}(\xi) w'(x_i)$$

For example, 若只取兩點 x_0 and x_1 則以 $n = 1$, $i = 0$ 有

$$\begin{aligned} f'(x) &= f[x_0, x_1] + \frac{1}{2}f''(\xi)\frac{d}{dx}[(x-x_0)(x-x_1)]|_{x=x_0} \\ &= f[x_0, x_1] + \frac{1}{2}f''(\xi)(x_0 - x_1) \end{aligned}$$

這即是 Eq.(4.21) 之一種型式當 $x_0 = x$ and $x_1 = x + h$.

2. The Second observation: 若選擇 x 使 $w'(x) = 0$, 則 Eq.(4.36) 變得更簡單. 例如, 當 $n = 1$時 $w(x)$ 是 a quadratic function(2次) 且 $w(x) = 0$ 在 $x = x_0$ 和 x_1. 因 a parabola (拋物線) 是 symmetric about its axis, 故 $w'((x_0 + x_1)/2) = 0$. Therefore, the resulting formula is

$$f'\left(\frac{x_0 + x_1}{2}\right) = f[x_0, x_1] - \frac{1}{8}(x_1 - x_0)^2\frac{d}{dx}f''(\xi)$$

3. 最後以 4 interpolation points x_0, x_1, x_2, and x_3 爲例. From eq.(8) in sec.4.1 with $n = 3$, the interpolation polynomial is

$$\begin{aligned} p_3(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &\quad + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2). \end{aligned}$$

Its derivative is

$$\begin{aligned} p_3'(x) &= f[x_0, x_1] + f[x_0, x_1, x_2](2x - x_0 - x_1) + f[x_0, x_1, x_2, x_3] \\ &\quad \times[(x - x_1)(x - x_2) + (x - x_0)(x - x_2) + (x - x_0)(x - x_1)] \end{aligned}$$

A useful special case occurs if $x_0 = x - h$, $x_1 = x + h$, $x_2 = x - 2h$, and $x_3 = x + 2h$, 則有

$$\begin{aligned} f'(x) &\approx \frac{1}{2h}[f(x + h) - f(x - h)] \\ &\quad - \frac{1}{12h}[f(x + 2h) - 2f(x + h) + 2f(x - h) - f(x - 2h)] \end{aligned} \tag{4.37}$$

此型即是可能被計算以: A principal term plus a correction or refining term.

Second-Derivative Formulas via Taylor Series

將 $f(x+h)$ 和 $f(x-h)$ 之 Taylor Series 相加得:

$$f(x+h)+f(x-h)=2f(x)+h^2f''(x)+2\left[\frac{1}{4!}h^4f^{(4)}(x)+\cdots\right]$$

重新安排上式得

$$f''(x)=\frac{1}{h^2}[f(x+h)-2f(x)+f(x-h)]+E$$

其中誤差級數爲

$$E=-2\left[\frac{1}{4!}h^2f^{(4)}(x)+\frac{1}{6!}h^4f^{(6)}(x)+\cdots\right]$$

相同於處理 Taylor's formula with a remainder 之方法可證得

$$E=-\frac{1}{12}h^2f^{(4)}(x)(\xi)$$

for some $\xi\in(x-h,\,x+h)$. 因此我們得二階導數之公式

$$f''(x)\approx\frac{1}{h^2}[f(x+h)-2f(x)+f(x-h)] \tag{4.38}$$

with error $O(h^2)$ $\sharp$