Chapter 3 Interpolation and Polynomial Approximation

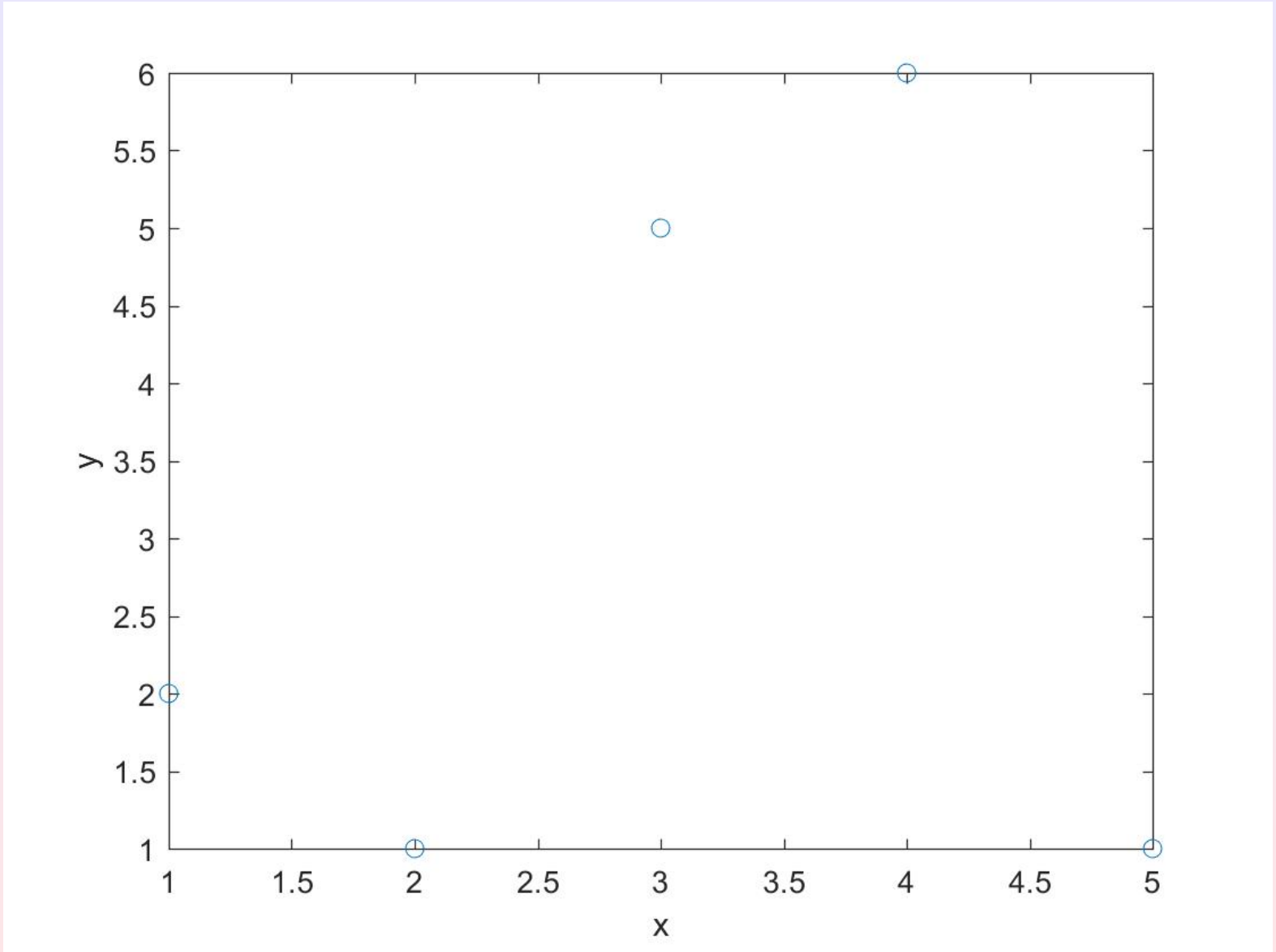
Baodong LIU
baodong@sdu.edu.cn

September 23, 2022

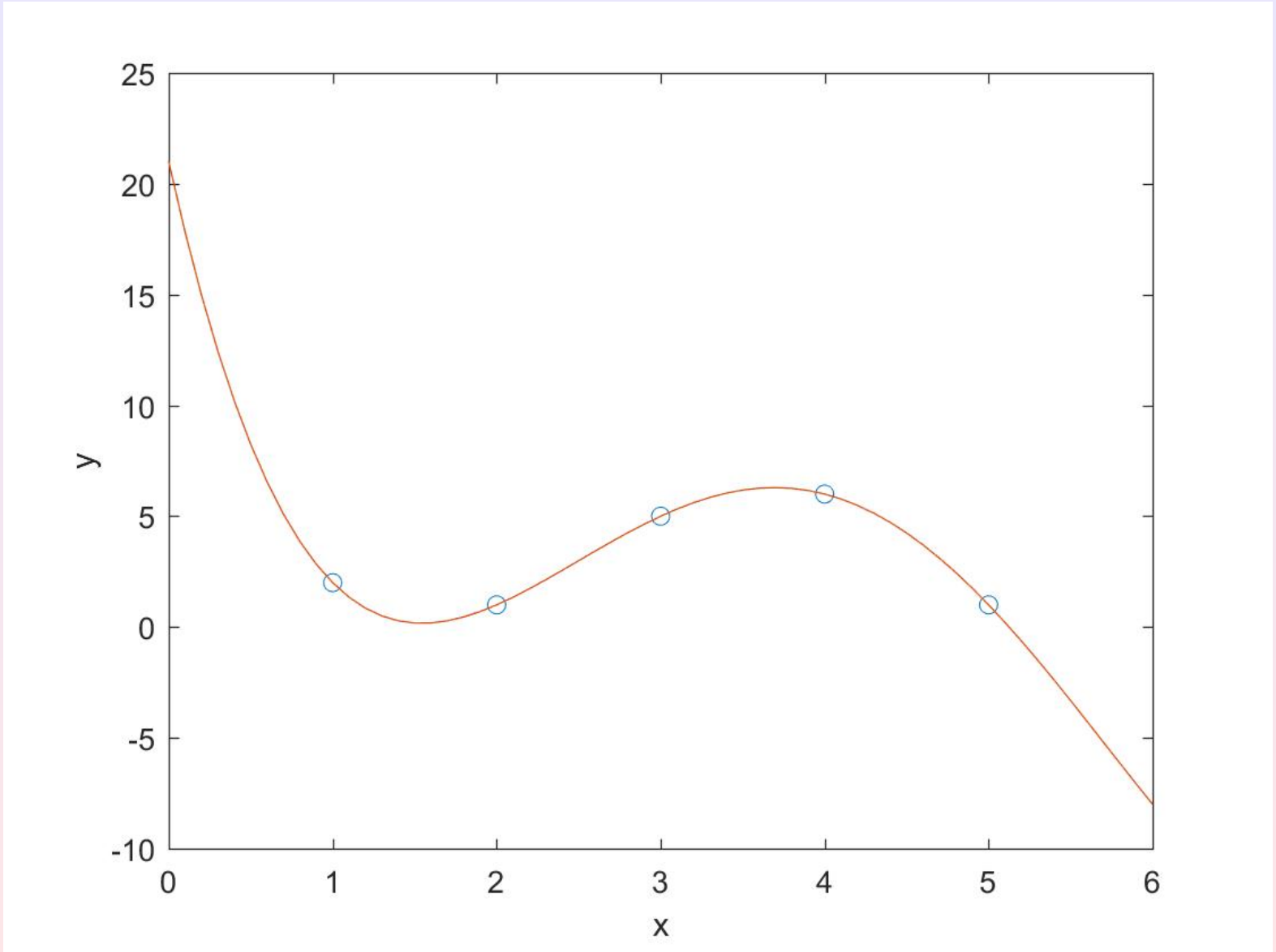
Main Contents:

- Interpolation and the Lagrange Polynomial;
- Divided Differences;
- Hermite Interpolation;
- Cubic Spline Interpolation.

Example for data modeling



Find a curve (or function) to fit the given points (or data)



Questions?

- Given some known monitoring data, such as

$$(x_0, f(x_0)), (x_1, f(x_1)), \cdots, (x_n, f(x_n)).$$

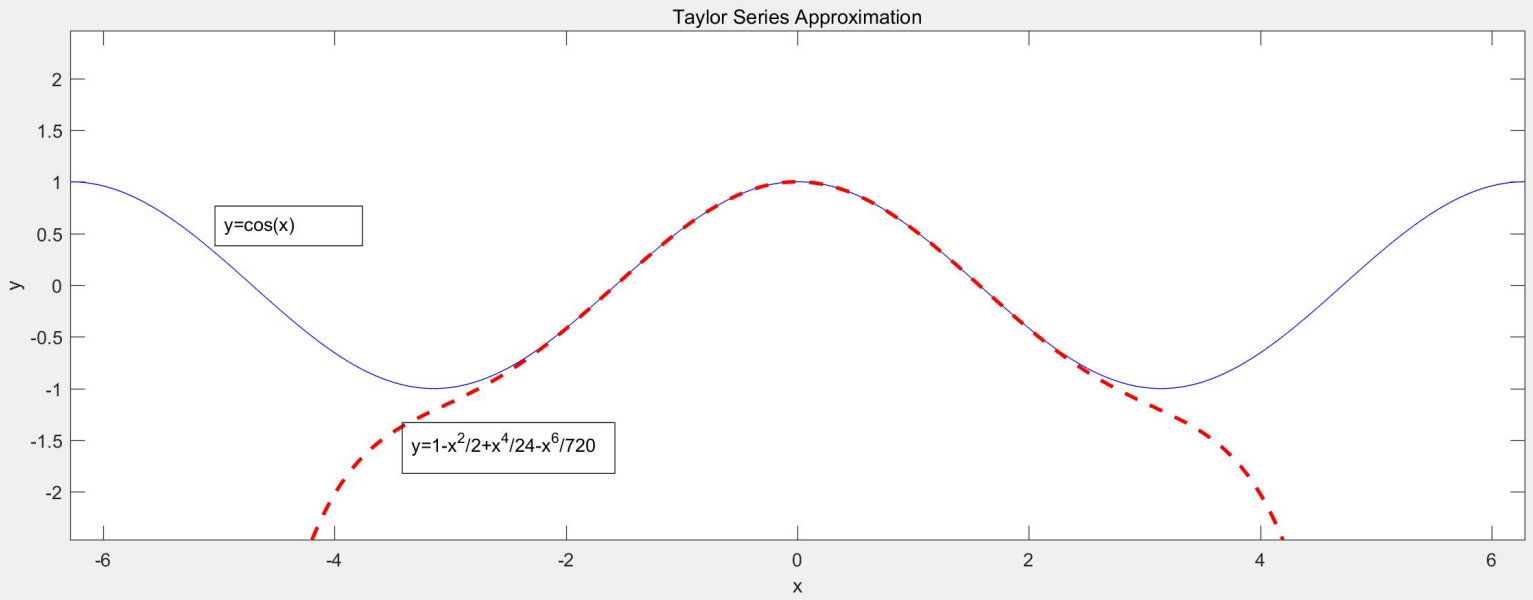
- Suppose these data are correct(no monitoring error exists).
- How could we predict the value of $f(x)$ at each non-monitoring point($x \neq x_i, i = 0, 1, 2, \cdots, n$)?
- For example:
  1. How can we evaluate $f(\bar{x}_i) = ?$ where

$$\bar{x}_i \in [x_{i-1}, x_i], i = 1, 2, \cdots, n-1.$$

  2. Can we predict the value $f(\bar{x})$ when $\bar{x} > x_n$?

- This kind of problem can be viewed as to **find a function** $f(x)$ that **fits** the given data.
- This process is called **Interpolation**.
- Does the fitting function exist?
  1. If so, how to construct a function to fit the given data?
  2. Is it unique?

以Taylor 多项式展开式为例，观察



$T_N(x) = x^4/24 - x^2/2 - x^6/720 + 1$

f(x) = cos(x)

N = 7 + -

a = 0

-2*pi < x < 2*pi

Help Reset Close

Theorem 4.1: Weierstrass Approximation Theorem

- Suppose f is defined and continuous on $[a, b]$.
- For each $\varepsilon > 0$, there exists a polynomial $P(x)$, defined on $[a, b]$, with the property that

$$|f(x) - P(x)| < \varepsilon, \forall\ x \in [a, b].\blacksquare$$

Question

- Does the Taylor polynomials suit for our case?
- If not, why?

3.1 Interpolation and the Lagrange polynomial

- Problem description:
  - Given some known monitoring data, such as

$$(x_0, f(x_0)), (x_1, f(x_1)), \cdots, (x_n, f(x_n)).$$

  - Find approximating polynomials $P(x)$, which must pass above given points, that is

$$P(x_i) = f(x_i), i = 0, 1, 2, \cdots, n.$$

- Basic concepts on interpolation method:
  - Such kind of problem is called **polynomial interpolation**.
  - For each $\bar{x} \in [\min\limits_{0 \le i \le n} x_i, \max\limits_{0 \le i \le n} x_i]$, to find the unknown $f(\bar{x})$, can be approximated by $P(\bar{x})$.

Case I: Linear Interpolating Polynomial

A polynomial of degree one that passes through the distinct points

$$(x_0, y_0) \quad \texttt{and} \quad (x_1, y_1)$$

with $x_0 \neq x_1$.

- Where y_0, y_1 maybe the corresponding monitoring values of an unknown function $f(x)$ at x_0, x_1.
- The problem is the same as approximating a given function $f(x)$ for which $f(x_0) = y_0$ and $f(x_1) = y_1$.

Case I: Construct Linear Interpolating Polynomial

- Define basis functions

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}, \quad L_1(x) = \frac{x - x_0}{x_1 - x_0}$$

- Then the Linear Interpolating Polynomial through (x_0, y_0) and (x_1, y_1) is:

$$P(x) = L_0(x)y_0 + L_1(x)y_1 = \frac{x - x_1}{x_0 - x_1}y_0 + \frac{x - x_0}{x_1 - x_0}y_1.$$

- It is clear that

$$L_0(x_0) = 1, L_0(x_1) = 0;$$

and

$$L_1(x_0) = 0, L_1(x_1) = 1.$$

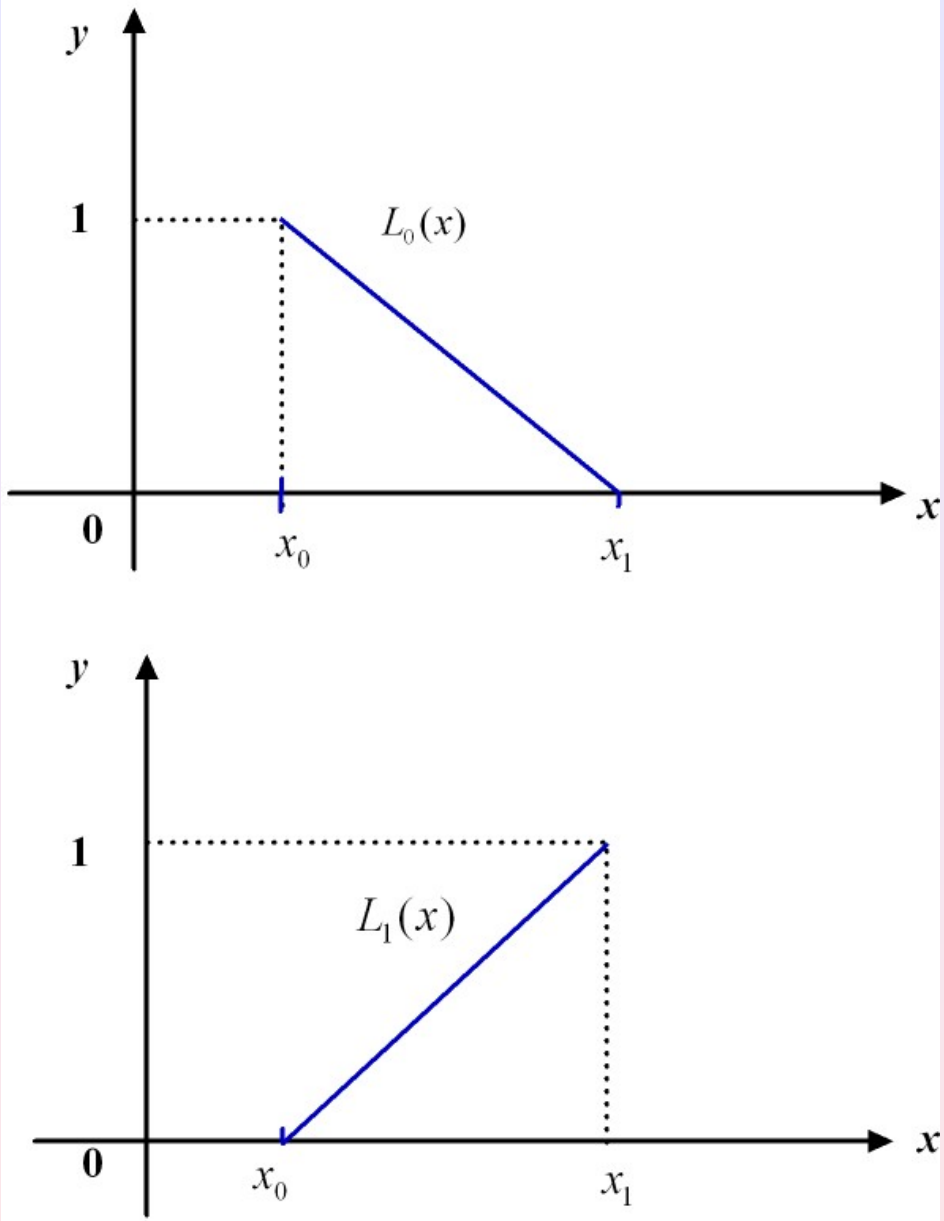


Fig.3-1 Linear Interpolation

Case II: the Quadratic Interpolation Polynomial

- Given three distinct points

$$(x_0, y_0), (x_1, y_1) \quad \text{and} \quad (x_2, y_2)$$

Or approximating a function $f(x)$ for which

$$f(x_0) = y_0, f(x_1) = y_1 \quad \text{and} \quad f(x_2) = y_2$$

- Determining a polynomial of degree two that passes through these three distinct points.
- Define the quadratic interpolation basis function using quotients

$$\begin{aligned} L_{2,0}(x) &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}, \\ L_{2,1}(x) &= \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}, \\ L_{2,2}(x) &= \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}. \end{aligned}$$

Quadratic Forms

- Then the basis functions satisfy the listed properties:

$$\begin{aligned} L_{2,0}(x_0) &= 1, L_{2,0}(x_1) = 0, L_{2,0}(x_2) = 0; \\ L_{2,1}(x_0) &= 0, L_{2,1}(x_1) = 1, L_{2,1}(x_2) = 0; \\ L_{2,2}(x_0) &= 0, L_{2,2}(x_1) = 0, L_{2,2}(x_2) = 1. \end{aligned}$$

- **Thus the Quadratic Interpoltaion Polynomial is**

$$\begin{aligned} P(x) &= L_{2,0}(x)y_0 + L_{2,1}(x)y_1 + L_{2,2}(x)y_2 \\ &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 \\ &\quad + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2, \end{aligned}$$

- It can be seen that $P(x_0) = y_0, P(x_1) = y_1, P(x_2) = y_2$, so $P(x)$ has the required properties also.

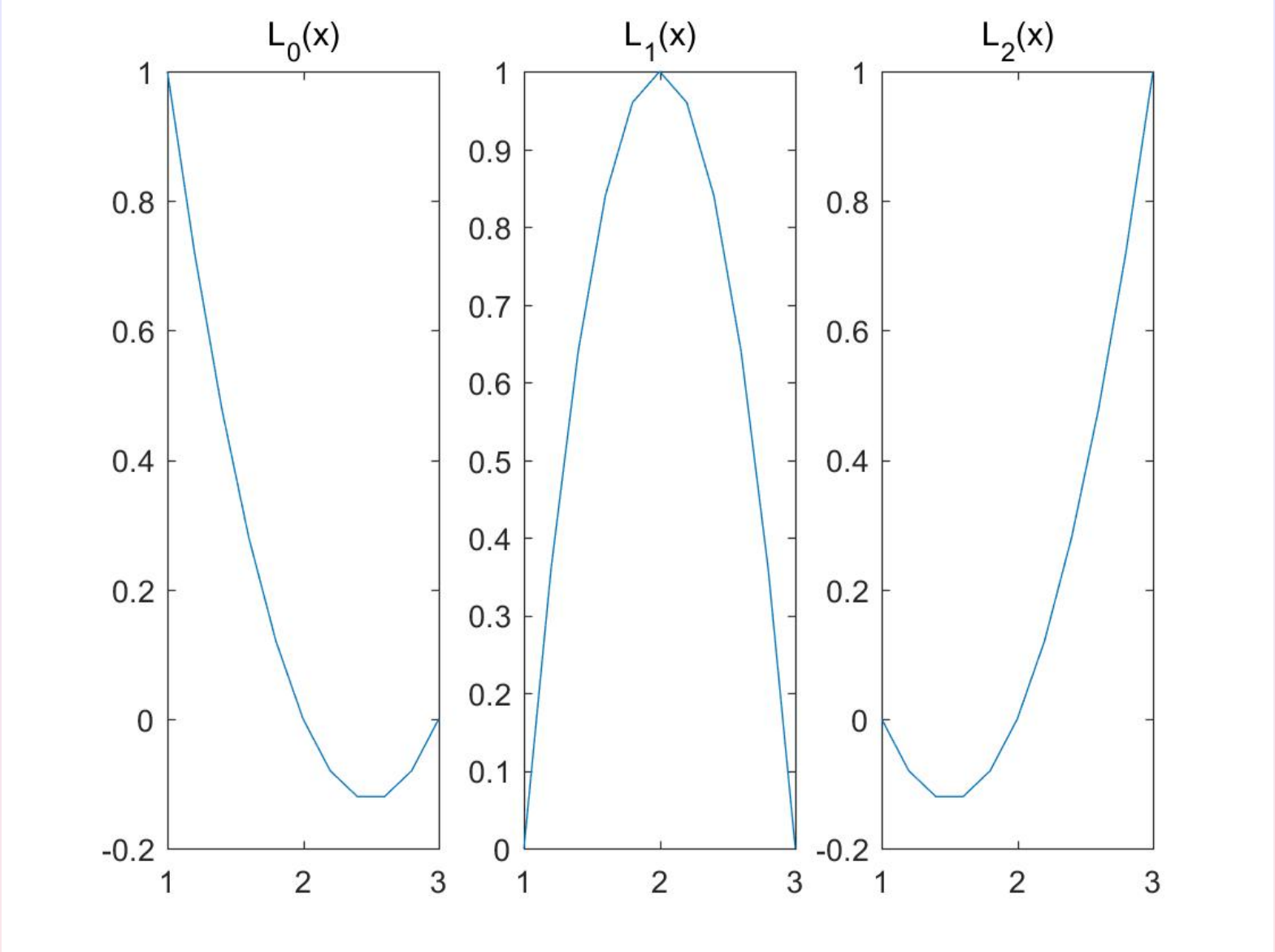


Fig.3-1 以$x = 1, 2, 3$ 为插值节点的二次插值基函数示意图

Case III: General Case

- To generalize the concept of linear interpolation, consider the construction of a polynomial of degree at most n that passes through the $n+1$ points

$$(x_0, f(x_0)), (x_1, f(x_1)), \cdots, (x_n, f(x_n)).$$

- How to construct the polynomial of degree at most n that passes through the $n+1$ points.

Genearl Case:

- For the general case we construct, for each $k = 0, 1, \cdots, n$, a quotient $L_{n,k}(x)$ be the basis function at x_k with the property that

$$L_{n,k}(x_i) = 0, \quad \text{when} \quad i \neq k \text{ and } \quad L_{n,k}(x_k) = 1.$$

- To satisfy $L_{n,k}(x_i) = 0$ for each $i \neq k$ requires that the numerator of $L_{n,k}(x)$ contain the term

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

- To satisfy $L_{n,k}(x_k) = 1$, the denominator of $L_{n,k}(x)$ must agree with former formula when it is evaluated at $x = x_k$.

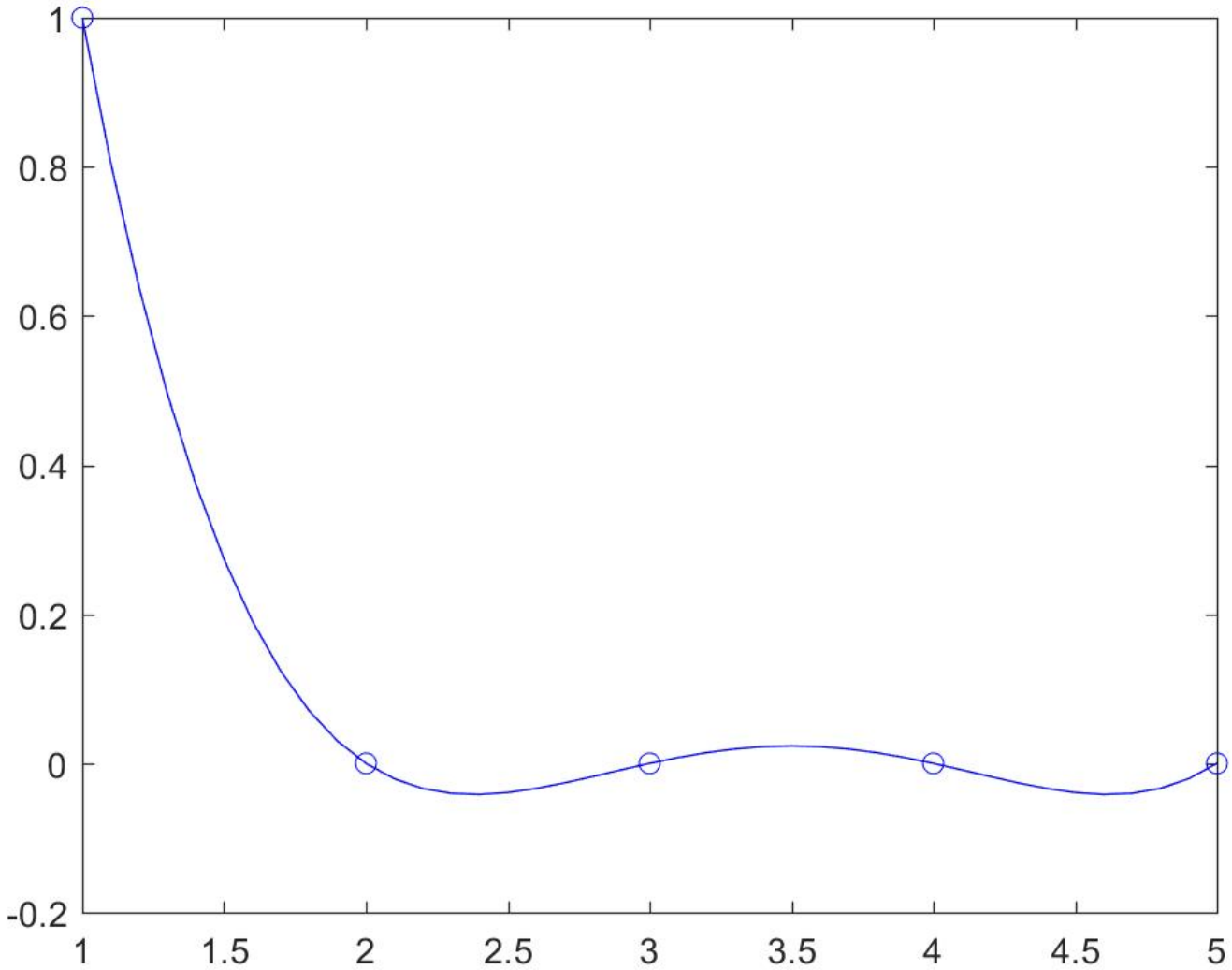


Figure: 3-3. (a) 以$x = 1, 2, 3, 4, 5$ 插值节点在$x = 1$ 的插值基函数

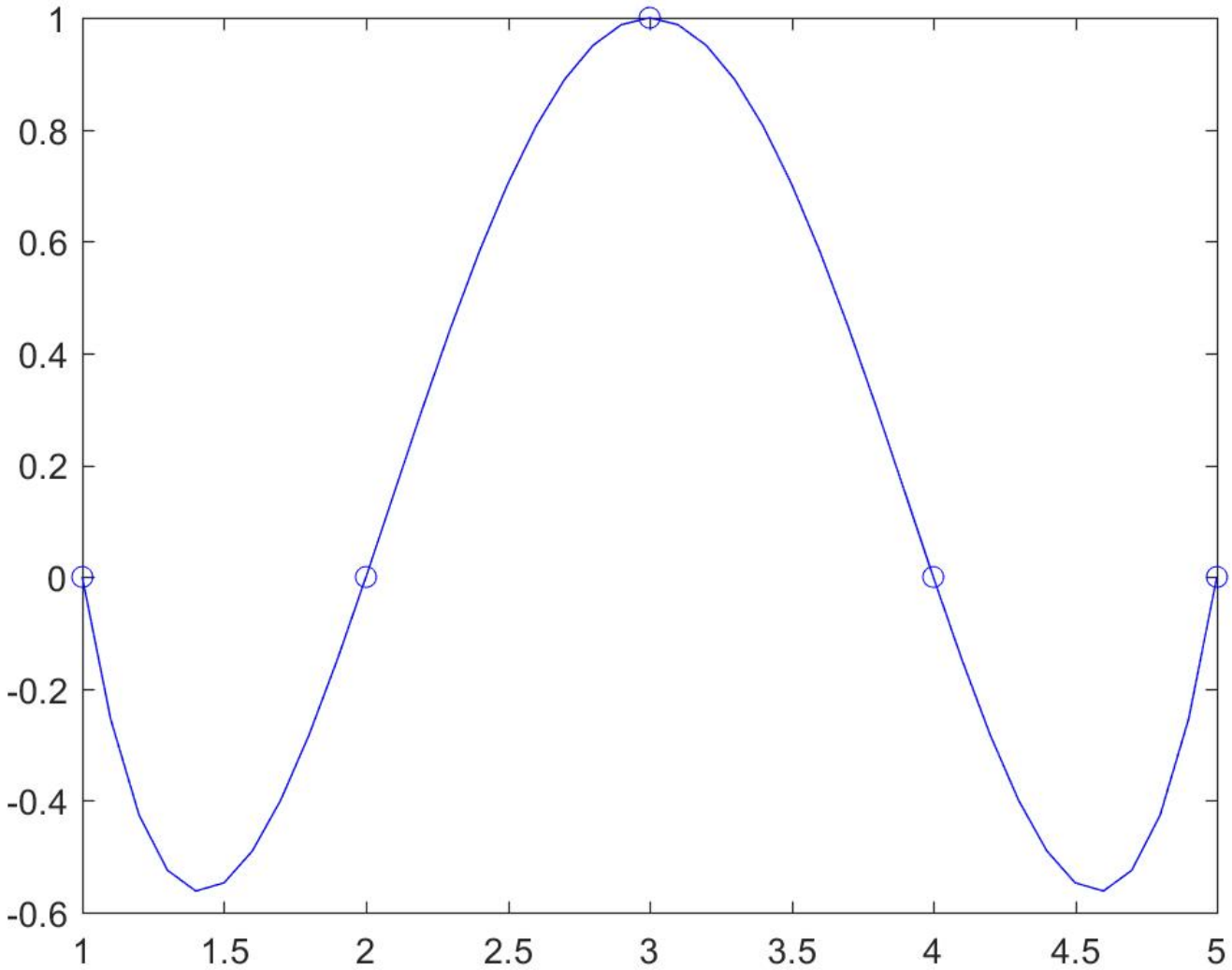


Figure: 3-3. (b) 以$x = 1, 2, 3, 4, 5$ 插值节点在$x = 3$ 的插值基函数

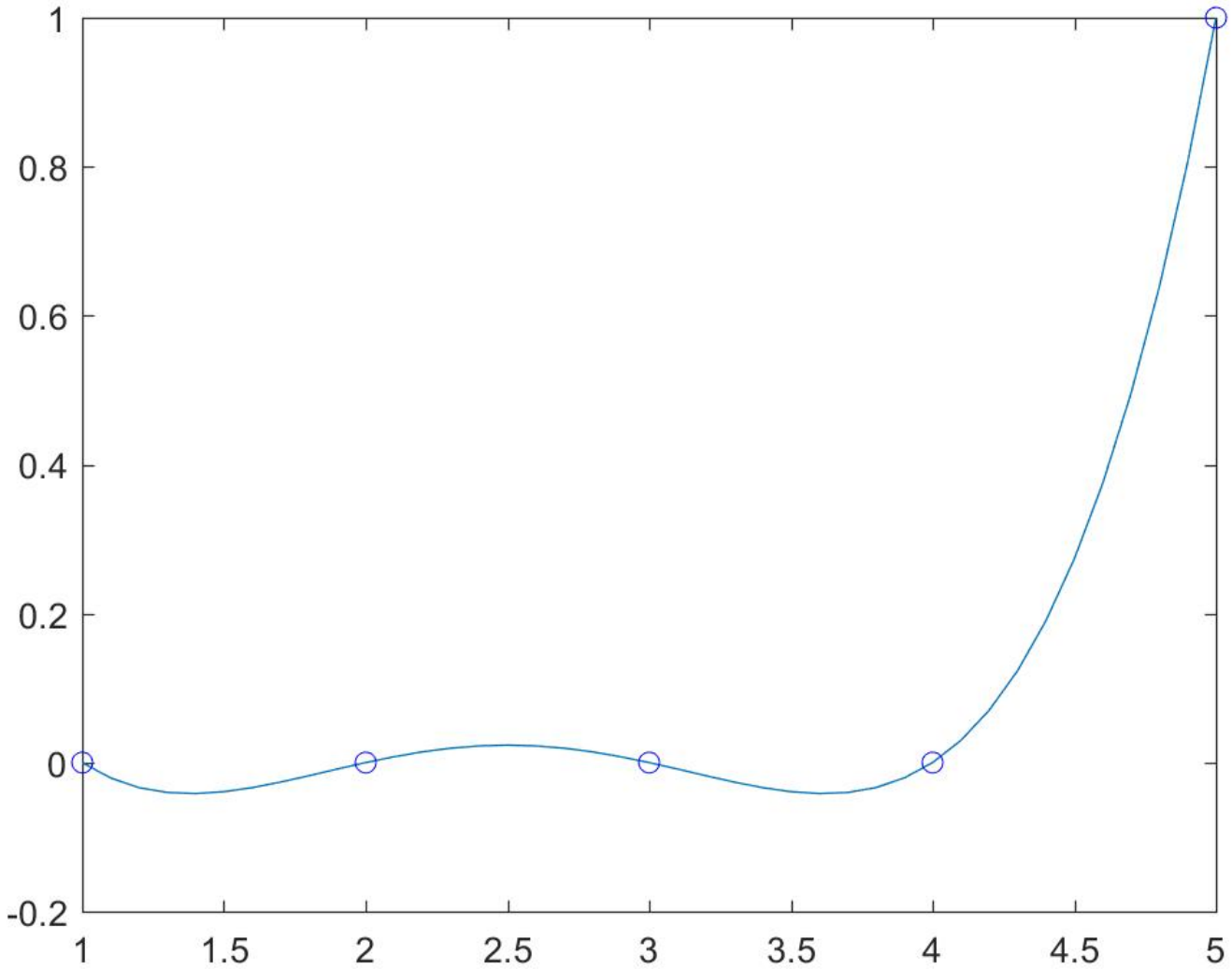


Figure: 3-3. (c) 以$x = 1, 2, 3, 4, 5$ 插值节点在$x = 5$ 的插值基函数

- That is

$$\begin{aligned} L_{n,k}(x) &= \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \\ &= \prod_{i=0,i\neq k}^{n} \frac{x-x_i}{x_k-x_i}. \end{aligned}$$

- The interpolating polynomial is easily described now, is called —— —— **the nth Lagrange Interpolating Polynomial** and is given by

$$\begin{aligned} P(x) &= \sum_{k=0}^{n} L_{n,k}(x) f(x_k) \\ &= \sum_{k=0}^{n} \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x\_x_n)} f(x_k) \\ &= \sum_{k=0}^{n} f(x_k) \prod_{i=0,i\neq k}^{n} \frac{x-x_i}{x_k-x_i}. \end{aligned}$$

- We will write $L_{n,k}(x)$ simply as $L_k(x)$ when there is no confusion as to its degree.

Theorem 3.2

- If $x_0, x_1, \cdots, x_n$ are $n+1$ distinct numbers
- $f(x)$ is a function whose values are given at these numbers
- there exists a **unique** polynomial $P(x)$ of degree at most n with the property that

$$P(x_i) = f(x_i), i = 0, 1, 2, \cdots, n$$

拉格朗日插值多项式逼近定理

Theorem 3.3

- Suppose $x_0, x_1, \cdots, x_n$ are distinct numbers in the interval $[a, b]$ and $f \in C^{n+1}[a, b]$.
- Then, for each $x \in [a, b]$, a number $\xi(x) \in (a, b)$ exists with

$$f(x) = P_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n)$$

where $P_n(x)$ is the interpolating polynomials of degree n defined above.■

Proof of Theorem 3.3

- If $x \neq x_k$ for any $k = 0, 1, \cdots, n$, define the function $R(t)$ for t in $[a, b]$ by

$$R(t) = (f(t) - P_n(t)) - [f(x) - P_n(x)] \prod_{i=0}^{n} \frac{t - x_i}{x - x_i},$$

- Since $f \in C^{n+1}[a, b]$, $P_n \in C^{\infty}[a, b]$, and $x \neq x_k$ for any k, it follows that $g \in C^{n+1}[a, b]$.
- For $t = x_k$ we have

$$\begin{aligned} R(x_k) &= (f(x_k) - P_n(x_k)) \\ &\quad -[f(x) - P_n(x)] \frac{(x_k - x_0) \cdots (x_k - x_n)}{(x - x_0) \cdots (x - x_n)}, \\ &= 0 - [f(x) - P_n(x)] \cdot 0, \\ &= 0. \quad \forall k = 0, 1, \cdots, n. \end{aligned}$$

- More ever, if $t = x$, we yield

$$\begin{aligned} R(x) &= (f(x) - P_n(x)) \\ &\quad -[f(x) - P_n(x)]\frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(x-x_0)(x-x_1)\cdots(x-x_n)}, \\ &= (f(x) - P_n(x)) - [f(x) - P_n(x)]\prod_{i=0}^{n}\frac{x-x_i}{x-x_i} \\ &= 0 \end{aligned}$$

- Thus, $R \in C^{n+1}[a,b]$ and R vanishes at the $n+2$ distinct numbers $x, x_0, x_1, \cdots, x_n$.

- By the Generalized Rolle's Theorem, there exists $\xi \in (a, b)$ for which $R^{n+1}(\xi) = 0$.
- So,

$$\begin{aligned} 0 &= R^{(n+1)}(\xi) \\ &= f^{(n+1)}(\xi) - P^{(n+1)}(\xi) - [f(x) - P(x)] \frac{(n+1)!}{\prod_{i=0}^{n}(x - x_i)} \end{aligned}$$

$$\Rightarrow f^{(n+1)}(\xi) - 0 - [f(x) - P(x)] \frac{(n+1)!}{\prod_{i=0}^{n}(x - x_i)} = 0$$

- Upon solving for $f(x)$, we have

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

Remarks:

1. The error estimation in this theorem is an important theoretical result.
2. It is used extensively for **Error bound estimations** in deriving **numerical differentiation and integration methods**.
3. For example, if $|f^{(n+1)}(\xi)| \leq M_{n+1}$, then
   - for the case $n = 1$,
   $$|f(x) - P_1(x)| = \frac{|f''(\xi)|}{2!}|(x - x_0)(x - x_1)| \leq \frac{M_2 h^2}{8}$$
   - for the case $n = 2$,
   $$|f(x) - P_2(x)| = \frac{|f^{(3)}(\xi)|}{3!}|(x - x_0)(x - x_1)(x - x_2)| \leq \frac{M_3 h^3}{9\sqrt{3}}$$
   - for the case $n = 3$,
   $$|f(x) - P_3(x)| = \frac{|f^{(4)}(\xi)|}{4!}|(x - x_0)(x - x_1)(x - x_2)(x - x_3)| \leq \frac{M_4 h^4}{24}$$

Remarks:

1. The error estimation in this theorem is an important theoretical result.
2. It is used extensively for **Error bound estimations** in deriving **numerical differentiation and integration methods**.
3. For example, if $|f^{(n+1)}(\xi)| \leq M_{n+1}$, then
   - for the case $n = 1$,
   $$|f(x) - P_1(x)| = \frac{|f''(\xi)|}{2!}|(x - x_0)(x - x_1)| \leq \frac{M_2 h^2}{8}$$
   - for the case $n = 2$,
   $$|f(x) - P_2(x)| = \frac{|f^{(3)}(\xi)|}{3!}|(x - x_0)(x - x_1)(x - x_2)| \leq \frac{M_3 h^3}{9\sqrt{3}}$$
   - for the case $n = 3$,
   $$|f(x) - P_3(x)| = \frac{|f^{(4)}(\xi)|}{4!}|(x - x_0)(x - x_1)(x - x_2)(x - x_3)| \leq \frac{M_4 h^4}{24}$$

Remarks:

There will meet some difficulties with using the Lagrange polynomials as described above

- First, the **error term** is difficult to apply, so the degree of the polynomial needed for desired accuracy is generally not known until computations are determined
- Second, the work done needed in calculating the approximation by second polynomial does not lessen the work done needed to calculate the third approximation, nor is the fourth approximation easier to obtain once the third approximation is known.

Definition 3.4

- Let f be a function defined at $x_0, x_1, x_2, \cdots, x_n$.
- Suppose that

$$m_1, m_2, ..., m_k$$

are k distinct integers with $0 \leq m_i \leq n$ for each i.

- The Lagrange polynomial that agrees with f at the k points

$$x_{m_1}, x_{m_2}, \cdots, x_{m_k},$$

is denoted

$$P_{m_1, m_2, \cdots, m_k}(x).$$

Theorem 3.5

Let f be defined at $x_0, x_1, x_2, \cdots, x_k$, and x_j and x_i be two distinct numbers in this set.
Then

$$P(x) = \frac{(x - x_j)P_{0,1,\cdots,j-1,j+1,\cdots,k}(x) - (x - x_i)P_{0,1,\cdots,i-1,i+1,\cdots,k}(x)}{(x_i - x_j)}$$

describes the kth Lagrange polynomial that interpolates f at the $k+1$ points $x_0, x_1, x_2, \cdots, x_k$.■

Proof of theorem 3.5

- For ease of notation, let

$$Q \equiv P_{0,1,\cdots,i-1,i+1,\cdots,k}$$

and

$$\hat{Q} \equiv P_{0,1,\cdots,j-1,j+1,\cdots,k}.$$

- Thus $Q(x)$ and $\hat{Q}(x)$ are both polynomials of degree $k-1$ or less, $P(x)$ is of degree at most k.
- If $0 \leq r \leq k$ and $r \neq i, j$ then $Q(x_r) = \hat{Q}(x_r) = f(x_r)$, so

$$\begin{aligned} P(x_r) &= \frac{(x_r - x_j)\hat{Q}(x_r) - (x_r - x_i)Q(x_r)}{x_i - x_j} \\ &= f(x_r). \end{aligned}$$

- Moreover,

$$\begin{aligned} P(x_i) &= \frac{(x_i - x_j)\hat{Q}(x_i) - (x_i - x_i)Q(x_i)}{x_i - x_j} \\ &= \frac{x_i - x_j}{x_i - x_j} f(x_i) \\ &= f(x_i). \end{aligned}$$

  and similarly, $P(x_j) = f(x_j)$.
- But, by the definition, $P_{0,1,\cdots,k}(x)$ is unique polynomial of degree at most k which with f at $x_0, x_1, \cdots, x_k$.
- Thus $P \equiv P_{0,1,\cdots,k}(x)$. ■■■

- Using this theorem, the interpolating polynomials can be generated recursively.
- Let $Q_{i,j}(x)$, for $0 \leq j \leq i$, denote the interpolating polynomial of degree j on the $j+1$ numbers $x_{i-j}, x_{i-j+1}, \cdots, x_{i-1}, x_i$.
- That is

$$Q_{i,j} = P_{i-j,i-j+1,\cdots,i-1,i}.$$

Neville's method

x_0	$P_0 = Q_{0,0}$				
x_1	$P_1 = Q_{1,0}$	$P_{0,1} = Q_{1,1}$			
x_2	$P_2 = Q_{2,0}$	$P_{1,2} = Q_{2,1}$	$P_{0,1,2} = Q_{2,2}$		
x_3	$P_3 = Q_{3,0}$	$P_{2,3} = Q_{3,1}$	$P_{1,2,3} = Q_{3,2}$	$P_{0,1,2,3} = Q_{3,3}$	
x_4	$P_4 = Q_{4,0}$	$P_{3,4} = Q_{4,1}$	$P_{2,3,4} = Q_{4,2}$	$P_{1,2,3,4} = Q_{4,3}$	$P_{0,1,2,3,4}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Process for Neville's method

- To compute 0th polynomial:

$$Q_{0,0} = f(x_0), Q_{1,0} = f(x_1), \cdots, Q_{k,0} = f(x_k), \cdots, Q_{n,0} = f(x_n)$$

- To compute 1th polynomial:

$$\begin{aligned}
Q_{1,1} &= \frac{(x - x_0)Q_{1,0} - (x - x_1)Q_{0,0}}{x_1 - x_0}, \\
Q_{2,1} &= \frac{(x - x_1)Q_{2,0} - (x - x_2)Q_{1,0}}{x_2 - x_1} \\
\cdots & \quad \cdots\cdots \\
Q_{n,1} &= \frac{(x - x_{n-1})Q_{n,0} - (x - x_n)Q_{n-1,0}}{x_n - x_{n-1}}
\end{aligned}$$

- To compute kth polynomial:

$$\begin{aligned} Q_{k,k} &= \frac{(x - x_{k-1})Q_{k,k-1} - (x - x_k)Q_{k-1,k-1}}{x_k - x_{k-1}}, \\ Q_{k+1,k} &= \frac{(x - x_k)Q_{k+1,k-1} - (x - x_{k+1})Q_{k,k-1}}{x_{k+1} - x_k} \\ \cdots & \quad \cdots\cdots \\ Q_{n,k} &= \frac{(x - x_{n-1})Q_{n,k-1} - (x - x_n)Q_{n-1,k-1}}{x_n - x_{n-1}} \end{aligned}$$

  for $k = 3, 4, \cdots, n$.

- To evaluate the interpolating polynomial P on the $n + 1$ distinct numbers $x_0, x_1, \cdots, x_n$ at the number x for the function f.

ALGORITHM 3.1: Neville's Iteration Interpolation Method

INPUT: numbers $x_0, x_1, \cdots, x_n$; values $f(x_0), f(x_1), \cdots, f(x_n)$ as the first column $Q(0,0), Q(1,0), \cdots, Q(n,0)$ of Q.

OUTPUT: the table Q with $P(x) = Q_{n,n}$.

Step 1 For $i = 1, 2, \cdots, n$

- for $j = 1, 2, \cdots, i$
- set $Q_{i,j} = \frac{(x - x_{i-j})Q_{i,j-1} - (x - x_i)Q_{i-1,j-1}}{x_i - x_{i-j}}$

Step 2 Output Q,STOP.

3.3 Divided Difference(差商或均差方法)

- **Idea:** Set $(x_0, f_0 = f(x_0))$ and $(x_1, f_1 = f(x_1))$, to draw a line pass these two points, we can define its line equation with point slope form

$$P_1(x) = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0).$$

- If we know and the values $f_0, f_1, f_2, \cdots, f_n$ of a function f at points $x_0, x_1, x_2, \cdots, x_n$, we hope to extend previous idea to the case of $n + 1$ points.
- Suppose that $P_n(x)$ is the nth Lagrange polynomial that agrees with the function f at the distinct numbers $x_0, x_1, x_2, \cdots, x_n$.

- The divided differences of f with respect to

$$x_0, x_1, x_2, \cdots, x_n$$

are derived to express $P_n(x)$ in the form

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

where $a_0, a_1, \cdots, a_n$ are undefined constants.

- To determine these constants, we can use the interpolation conditions

$$f(x_i) = P(x_i) = f_i, i = 0, 1, \cdots, n.$$

- That is

  Let $x = x_0$,

$$\begin{aligned} P_n(x_0) &= a_0 = f_0, \\ a_0 &= f_0. \end{aligned}$$

  Let $x = x_1$,

$$\begin{aligned} P_n(x_1) &= a_0 + a_1(x_1 - x_0) = f_1, \\ a_1 &= \frac{f_1 - f_0}{x_1 - x_0}. \end{aligned}$$

  Let $x = x_2$,

$$\begin{aligned} P_n(x_2) &= a_0 + a_1(x_1 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = f_2 \\ a_2 &= \frac{\frac{f_2 - f_0}{x_2 - x_0} - \frac{f_1 - f_0}{x_1 - x_0}}{(x_2 - x_1)} = \frac{\frac{f_2 - f_1}{x_2 - x_1} - \frac{f_1 - f_0}{x_1 - x_0}}{x_2 - x_0} \\ \cdots &= \cdots \end{aligned}$$

To simplify the notation, some useful definitions are defined:

- **The zeroth divided difference**(零阶均差) of the function f with respect to x_i, denoted $f[x_i]$, is simply the value of f at x_i:

$$f[x_i] = f_i$$

- The remaining divided differences are defined inductively; **the first divided difference(一阶均差)** of f with respect to x_i and x_{i+1} is denoted $f[x_i, x_{i+1}]$ and defined as

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

- **The second divided difference** $f[x_i, x_{i+1}, x_{i+2}]$ is defined as

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

- Similarly, after **the** $(k-1)$**st divided differences**:

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

and

$$f[x_{i+1}, x_{i+2}, \cdots, x_{i+k-1}, x_{i+k}],$$

have been determined,

- **the** k**th divided difference** relative to

$$x_i, x_{i+1}, x_{i+2}, \cdots, x_{i+k}$$

is given by

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

- With this notation, Polynomial Equation can be re-expressed as

$$\begin{aligned}
a_0 &= f[x_0], \\
a_1 &= f[x_0, x_1]; \\
a_2 &= f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}; \\
&\cdots \quad \cdots \quad \cdots \\
a_n &= f[x_0, x_1, \cdots, x_n] \\
&= \frac{f[x_1, \cdots, x_n] - f[x_0, x_1, \cdots, x_{n-1}]}{x_n - x_0}.
\end{aligned}$$

Newton's interpolatory divided difference formula–牛顿均差插值公式

- $P_n(x)$ can be rewritten as

$$\begin{aligned} P_n(x) &= f[x_0] \\ &\quad + f[x_0, x_1](x - x_0) \\ &\quad + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &\quad + \cdots \\ &\quad + f[x_0, x_1, \cdots, x_n](x - x_0)(x - x_1)\cdots(x - x_{n-1}) \\ &= f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \cdots, x_k](x - x_0)(x - x_1)\cdots(x - x_{k-1}). \end{aligned}$$

- This equation is known as **Newton's interpolatory divided difference formula–牛顿均差插值公式**.

Table of the divided difference computation

x_k	$f(x_k)$	First Divided Difference	Second Divided Difference	$\cdots$ $\cdots$	The nth Divided Difference
x_0	$f[x_0]$				
x_1	$f[x_1]$	$f[x_0, x_1]$			
x_2	$f[x_2]$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$		
x_3	$f[x_3]$	$f[x_2, x_3]$	$f[x_1, x_2, x_3]$	$f[x_0, x_1, x_2, x_3]$	
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	
x_n	$f[x_n]$	$f[x_{n-1}, x_n]$	$f[x_{n-2}, x_{n-1}, x_n]$	$\cdots$	$f[x_0, x_1, \cdots, x_n]$

ALGORITHM 3.2 (Newton's Interpolatory Divided-Difference Formula)

INPUT numbers $x_0, x_1, \cdots, x_n$; values $f(x_0), f(x_1), \cdots, f(x_n)$ as $F_{0,0}, F_{1,0}, \cdots, F_{n,0}$

OUTPUT the numbers $F_{0,0}, F_{1,1}, \cdots, F_{n,n}$ where

$$P(x) = \sum_{i=0}^{n} F_{i,i} \prod_{j=0}^{i-1} (x - x_j).$$

Step 1
- For $i = 1, 2, \cdots, n$,
- For $j = 1, 2, \cdots, i$,
- $F_{i,j} = (F_{i,j-1} - F_{i-1,j-1})/(x_i - x_{i-j})$.

Step 2 OUTPUT $F_{0,0}, F_{1,1}, \cdots, F_{n,n}$, Stop.

- **Note** that $F_{i,i}$ is $f[x_0, x_1, \cdots, x_i]$.

THEOREM 3.6

- Suppose that $f \in C^n[a, b]$ and $x_0, x_1, \cdots, x_n$ are distinct numbers in $[a, b]$.
- Then a number ξ exists in (a, b) with

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

Proof of theorem 3.6

- Let
$$R(x) = f(x) - P_n(x)$$
- By the definition, we know that
$$f(x_i) = P_n(x_i), i = 0, 1, \cdots, n,$$
thus $g(x)$ has $n+1$ distinct zeros in $[a, b]$.
- Then, from the generalized Rolle's Theorem, there exists a number $\xi \in (a, b)$ with
$$R^{(n)}(\xi) = 0.$$

- Then, from the generalized Rolle's Theorem, there exists a number $\xi \in (a, b)$ with

$$R^{(n)}(\xi) = 0.$$

- Since $f \in C^n[a, b]$ and $P_n(x)$ is a polynomial of degree n.
- Thus we can derive that

$$0 = f^{(n)}(\xi) - P_n^{(n)}(\xi) = f^{(n)}(\xi) - f[x_0, x_1, \cdots, x_n]n!,$$

- Rewrite this equation, we get

$$f[x_0, x_1, \cdots, x_n] = \frac{f^{(n)}(\xi)}{n!}. \qquad \blacksquare\blacksquare\blacksquare$$

Simple form with equal spacing

- Introducing the notation

$$h = x_{i+1} - x_i, i = 0, 1, \cdots, n-1$$

and

$$x = x_0 + sh,$$

thus the difference $x - x_i$ can be written as

$$x - x_i = (s - i)h.$$

Simple form with equal spacing

- So

$$
\begin{aligned}
P_n(x) &= P_n(x_0 + sh) \\
&= f[x_0] \\
&\quad + shf[x_0, x_1] \\
&\quad + s(s-1)h^2 f[x_0, x_1, x_2] + \cdots + \\
&\quad + s(s-1)\cdots(s-n+1)h^n f[x_0, x_1, \cdots, x_n] \\
&= \sum_{k=0}^{n} s(s-1)\cdots(s-k+1)h^k f[x_0, x_1, \cdots, x_k].
\end{aligned}
$$

Newton forward divided-difference formula

- Using **binomial**(二项式)-coefficient notation,

$$\left(\begin{array}{c} s \\ k \end{array} \right) = \frac{s(s-1)\cdots(s-k+1)}{k!},$$

- We can express compactly as

$$\begin{aligned} P_n(x) &= P_n(x_0 + sh) \\ &= \sum_{k=0}^{n} \left(\begin{array}{c} s \\ k \end{array} \right) k! h^k f[x_0, x_1, \cdots, x_k] \end{aligned}$$

- This formula is called the **Newton forward divided-difference formula**.

Another form: Newton forward-difference formula

- Making use of the **forward difference** notation Δ introduced in **Aitken's Δ^2 method**:

$$\Delta f(x_i) = f(x_{i+1}) - f(x_i).$$

- With this notation,

$$\begin{aligned} f[x_0, x_1] &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1}{h}\Delta f(x_0), \\ f[x_0, x_1, x_2] &= \frac{1}{2h}\frac{\Delta f(x_1) - \Delta f(x_0)}{h} = \frac{1}{2h^2}\Delta^2 f(x_0), \end{aligned}$$

- In general

$$f[x_0, x_1, x_2, \cdots, x_k] = \frac{1}{k!h^k}\Delta^k f(x_0),$$

- Rewrite Newton Forward Divided Difference Formula as **Newton Forward Difference Formula**:

$$P_n(x) = P_n(x_0 + sh) = \sum_{k=0}^{n} \binom{s}{k} \Delta^k f(x_0)$$

Newton backward divided-difference formula

- If the interpolating nodes are reordered as

$$x_n, x_{n-1}, \cdots, x_0,$$

a formula similar to previous results is

$$\begin{aligned} P_n(x) = \quad & f[x_n] \\ & + f[x_n, x_{n-1}](x - x_n) \\ & + f[x_n, x_{n-1}, x_{n-2}](x - x_n)(x - x_{n-1}) \\ & + \cdots \\ & + f[x_n, x_{n-1}, \cdots, x_0](x - x_n)(x - x_{n-1}) \cdots (x - x_1), \end{aligned}$$

- If the nodes are equally spaced with $x = x_n + sh$ and

$$x_i = x_n - (n-i)h, \quad i = 0, 1, \cdots, n,$$

- Then

$$\begin{aligned} P_n(x) &= P_n(x_n + sh) \\ &= f[x_n] \\ &\quad + shf[x_n, x_{n-1}] \\ &\quad + s(s+1)h^2 f[x_n, x_{n-1}, x_{n-2}] \\ &\quad + \cdots \\ &\quad + s(s+1)\cdots(s+n-1)h^n f[x_n, x_{n-1}, \cdots, x_0], \end{aligned}$$

- This form is called the **Newton backward divided - difference formula**.

DEFINITION 3.7

Given the sequence $\{P_n\}_{n=0}^{\infty}$, define the backward difference ∇P_n (read nabla P_n) by

$$\nabla P_n = P_n - P_{n-1}, n \geq 1$$

Higher powers are defined recursively by

$$\nabla^k P_n = \nabla(\nabla^{k-1} P_n), k \geq 2.\blacksquare$$

This Definition implies that

$$\begin{aligned} f[x_n, x_{n-1}] &= \frac{1}{h}\nabla f(x_n), \\ f[x_n, x_{n-1}, x_{n-2}] &= \frac{1}{2h^2}\nabla^2 f(x_n), \\ \cdots &= \cdots \\ f[x_n, x_{n-1}, \cdots, x_{n-k}] &= \frac{1}{k!h^k}\nabla^k f(x_n). \end{aligned}$$

- Consequently,

$$\begin{aligned} P_n(x) &= P_n(x_n + sh) \\ &= f[x_n] + s\nabla f(x_n) + \frac{s(s+1)}{2}\nabla^2 f(x_n) + \cdots \\ &+ \frac{s(s+1)\cdots(s+n-1)}{n!}\nabla^n f(x_n), \end{aligned}$$

- We extend the binomial coefficient notation to include all real values of s by letting

$$\begin{pmatrix} -s \\ k \end{pmatrix} = \frac{-s(-s-1)\cdots(-s-k+1)}{k!}$$

$$= (-1)^k \frac{s(s+1)\cdots(s+k-1)}{k!},$$

This gives the following result
—**Newton Backward-Difference Formula**:

$$\begin{aligned} P_n(x) &= f(x_n) + (-1)^1 \binom{-s}{1} \nabla f(x_n) \\ &\quad + (-1)^2 \binom{-s}{2} \nabla^2 f(x_n) \\ &\quad + \cdots + (-1)^n \binom{-s}{n} \nabla^n f(x_n) \\ &= \sum_{k=0}^{n} (-1)^k \binom{-s}{k} \nabla^k f(x_n). \end{aligned}$$

Centered-difference formulas

- The Newton formulas are not appropriate for approximating a value x that lies near the center of the table since employing either the backward or forward method in such a way that the highest-order difference is involved will not allow x_0 to be close to x.
- A number of divided-difference formulas are available in this instance, each of which has situations when it can be used to maximum advantage. These methods are known as **centered-difference formulas**.
- There are a number of such methods, but we will present only one, **Stirling's method**.

- For **the centered-difference formulas** we choose x_0 near the point being approximated and label the nodes directly below x_0 as

$$x_1, x_2, \cdots,$$

and those directly above as

$$x_{-1}, x_{-2}, \cdots.$$

- With this convention, and if n is odd, let $n = 2m + 1$, then the nodes can be written as following:

$$x_{-m-1}, x_{-m}, \cdots, x_{-1}, x_0, x_1, \cdots, x_m, x_{m+1}.$$

With this convention, Stirling's formula can be given by the following formula, when $n = 2m + 1$.

$$
\begin{aligned}
P_n(x) &= P_{2m+1}(x) \\
&= f[x_0] + \frac{sh}{2}(f[x_{-1}, x_0] + f[x_0, x_1]) + \\
&\quad + s^2 h^2 f[x_{-1}, x_0, x_1] \\
&\quad + \frac{s(s^2-1)h^3}{2}(f[x_{-2}, x_{-1}, x_0, x_1] + f[x_{-1}, x_0, x_1, x_2]) + \\
&\quad + \cdots + \\
&\quad + s^2(s^2-1)\cdots(s^2-(m-1)^2)h^{2m} f[x_{-m}, \cdots, x_0, \cdots, x_m] \\
&\quad + \frac{s^2(s^2-1)\cdots(s^2-m^2)h^{2m+1}}{2}(f[x_{-m-1}, \cdots, x_0, \cdots, x_m] \\
&\quad + f[x_{-m}, x_{-m+1}, \cdots, x_0, x_1, \cdots, x_{m+1}]).
\end{aligned}
$$

Similarity, if $n = 2m$, then

$$
\begin{aligned}
P_n(x) \quad = \quad & P_{2m}(x) = f[x_0] \\
& + \frac{sh}{2}(f[x_{-1}, x_0] + f[x_0, x_1]) + \\
& + s^2 h^2 f[x_{-1}, x_0, x_1] \\
& + \frac{s(s^2-1)h^3}{2}(f[x_{-2}, x_{-1}, x_0, x_1] + f[x_{-1}, x_0, x_1, x_2]) \\
& + \cdots \\
& + s^2(s^2-1)\cdots(s^2-(m-1)^2)h^{2m} f[x_{-m}, \cdots, x_0, \cdots, x_m],
\end{aligned}
$$

3.4 Hermite Interpolation–厄米特插值

- Given two distinct numbers: x_0, x_1 with the function values
$$f_0 = f(x_0), f_1 = f(x_1).$$
- Suppose their derivative
$$f'(x_0) = m_0, f'(x_1) = m_1$$
also known.
- Construct a **Osculating Polynomial**(–密切多项式) $H_3(x)$, such that
$$H_3(x_0) = f_0 = f(x_0), H_3(x_1) = f_1 = f(x_1)$$
and
$$H_3'(x_0) = m_0 = f'(x_0), H_3'(x_1) = m_1 = f'(x_1)$$

Construction Method (Base functions):

- To construct Hermite Interpolation Base function, which is a polynomial of degree at most three

$$H_{2,0}(x), H_{2,1}(x), \hat{H}_{2,0}(x), \hat{H}_{2,1}(x).$$

Such that

$$H_{2,0}(x_0) = 1, H_{2,0}(x_1) = 0, H'_{2,0}(x_0) = 0, H'_{2,0}(x_1) = 0$$

$$H_{2,1}(x_0) = 0, H_{2,1}(x_1) = 1, H'_{2,1}(x_0) = 0, H'_{2,1}(x_1) = 0$$

$$\hat{H}_{2,0}(x_0) = 0, \hat{H}_{2,0}(x_1) = 0, \hat{H}'_{2,0}(x_0) = 1, \hat{H}'_{2,0}(x_1) = 0$$

$$\hat{H}_{2,1}(x_0) = 0, \hat{H}_{2,1}(x_1) = 0, \hat{H}'_{2,1}(x_0) = 0, \hat{H}'_{2,1}(x_1) = 1$$

- For $H_{2,0}(x)$, let

$$H_{2,0}(x) = (a + b(x - x_0))\frac{(x - x_1)^2}{(x_0 - x_1)^2}$$

- As $H_{2,0}(x_0) = 1, H'_{2,0}(x_0) = 0$, implies that

$$a = 1, b = -\frac{2}{x_0 - x_1}$$

- That is

$$H_{2,0}(x) = \left(1 + 2\frac{x - x_0}{x_1 - x_0}\right)\left(\frac{x - x_1}{x_0 - x_1}\right)^2.$$

- Similarly

$$\hat{H}_{2,0}(x) = (x - x_0)\left(\frac{x - x_1}{x_0 - x_1}\right)^2.$$

- $$H_{2,1}(x) = \left(1 + 2\frac{x - x_1}{x_0 - x_1}\right)\left(\frac{x - x_0}{x_1 - x_0}\right)^2.$$

- $$\hat{H}_{2,1}(x) = (x - x_1)\left(\frac{x - x_0}{x_1 - x_0}\right)^2.$$

- So the Hermite Polynomials can be written as

$$H_3(x) = H_{2,0}(x)f_0 + H_{2,1}(x)f_1 + \hat{H}_{2,0}(x)m_0 + \hat{H}_{2,1}(x)m_1$$

General Case

- Given $n+1$ distinct numbers

$$x_0, x_1, \cdots, x_n$$

and non-negative integers

$$k_0, k_1, \cdots, k_n,$$

- To construct the osculating polynomial approximating a function $f \in C^m[a, b]$, where

$$m = \max\{k_0, k_1, \cdots, k_n\}$$

- The polynomial of least degree with the property that it agrees with the function f and all its derivatives of order less than or equal to k_i at x_i for each $i = 0, 1, \cdots, n$.

- The degree of this **osculating polynomial** is at most

$$K = \sum_{i=0}^{n} k_i + n$$

- The number of conditions to be satisfied is

$$\sum_{i=0}^{n} (k_i + 1) = \sum_{i=0}^{n} k_i + (n + 1).$$

- A polynomial of degree K has $K + 1$ coefficients that can be used to satisfy these conditions.

DEFINITION 3.8

- Let

$$x_0, x_1, \cdots, x_n$$

be $n+1$ distinct numbers in $[a, b]$.

- k_i be a nonnegative integer associated with x_i for $i = 0, 1, \cdots, n$
- Suppose that $f \in C^m[a, b]$ and

$$m = \max\{k_0, k_1, \cdots, k_n\}.$$

- The **osculating polynomial** approximating f is the polynomial $P(x)$ of least degree such that

$$\frac{\mathrm{d}^s P(x_i)}{\mathrm{d}x^s} = \frac{\mathrm{d}^s f(x_i)}{\mathrm{d}x^s}$$

for each $i = 0, 1, \cdots, n$ and $s = 0, 1, \cdots, k_i$.■

Note that:

- when $n = 0$, the osculating polynomial approximating f is simply the kth Taylor polynomial for f at x_0.
- When $k_i = 0$ for each i, the osculating polynomial is the nth Lagrange polynomial interpolating f on $x_0, x_1, \cdots, x_n$.
- The case when $k_i = 1$ for each $i = 0, 1, \cdots, n$ gives a class called the **Hermite polynomials**.
- For a given function f, these polynomials agree with f at $x_0, x_1, \cdots, x_n$.
- In addition, if their first derivatives agree with those of f, they have the same "shape" as the function at $(x_i, f(x_i))$ in the sense that the **tangent lines** to the polynomial and to the function agree.

THEOREM 3.9

- If $f \in C^1[a, b]$ and $x_0, x_1, \cdots, x_n \in [a, b]$ are distinct, the unique polynomial of least degree agreeing with f and f' at $x_0, x_1, \cdots, x_n$ 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

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

THEOREM 3.9– continued

- where $L_{n,j}(x)$ denotes the jth Lagrange coefficient polynomial of degree n in previous sections

$$\begin{aligned} L_{n,j}(x) &= \frac{(x-x_0)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)} \\ &= \prod_{i=0,i\neq j}^{n} \frac{x-x_i}{x_j-x_i}. \\ L'_{n,j}(x_j) &= \sum_{i=0,i\neq j}^{n} \frac{1}{x_j-x_i}. \end{aligned}$$

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

$$f(x) = H_{2n+1}(x) + \frac{(x-x_0)^2(x-x_1)^2\cdots(x-x_n)^2}{(2n+2)!} f^{2n+2}(\xi),$$

for some ξ with $a < \xi < b$.■

Proof of THEOREM 3.9:

- First recall that

$$L_{n,j}(x_i) = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases}$$

- Hence when $i \neq j$.

$$H_{n,j}(x_i) = 0, \text{and } \hat{H}_{n,j}(x_i) = 0.$$

- Whereas

$$H_{n,i}(x_i) = [1 - 2(x_i - x_i)L'_{n,i}(x_i)] \cdot 1 = 1.$$

and

$$\hat{H}_{n,i}(x_i) = (x_i - x_i) \cdots 1^2 = 0.$$

- As a consequence,

$$\begin{aligned} H_{2n+1}(x_i) &= \sum_{j=0, j\neq i}^{n} f(x_j)\cdot 0 + f(x_i)\cdot 1 + \sum_{j=0}^{n} f'(x_j)\cdot 0 \\ &= f(x_i), \end{aligned}$$

  so $H_{2n+1}(x)$ agree with f at $x_0, x_1, \cdots x_n$.
- To show the agreement of $H'_{2n+1}(x)$ with f' at the nodes, first note that $L_{n,j}(x)$ is a factor of $H'_{n,j}(x)$. so $H'_{2n+1}(x_i) = 0$ when $i \neq j$.

- In addition, when $i = j$,

$$\begin{aligned} H'_{n,i}(x_i) &= -2L'_{n,i}(x_i) \cdot L^2_{n,i}(x_i) \\ &\quad +[1 - 2(x_i - x_i)L'_{n,i}(x_i)]2L_{n,i}(x_i)L'_{n,i}(x_i) \\ &= -2L'_{n,j}(x_i) + 2L'_{n,i}(x_i) = 0. \end{aligned}$$

- Finally,

$$\begin{aligned} \hat{H}'_{n,j} &= L^2_{n,j}(x_i) + (x_i - x_j)2L_{n,j}(x_i)L'_{n,j}(x_i) \\ &= L_{n,j}(x_i)[L_{n,j}(x_i) + 2(x_i - x_j)L'_{n,j}(x_i)], \end{aligned}$$

So $\hat{H}'_{n,j}(x_i) = 0$ if $i \neq j$ and $\hat{H}'_{n,i}(x_i) = 1$.

- Combining these facts we have

$$\begin{aligned} H'_{2n+1}(x_i) &= \sum_{j=0}^{n} f(x_j) \cdot 0 \\ &\quad + \sum_{j=0, j\neq i}^{n} f'(x_j) \cdot 0 + f'(x_i) \cdot 1 \\ &= f'(x_i). \end{aligned}$$

- Therefore, $H_{2n+1}(x)$ agrees with f and $H'_{2n+1}(x)$ with f' at $x_0, x_1, \cdots, n$.
- The uniqueness and error are as homework.■

高次插值多项式的**龙格现象**

- 插值节点的数目决定了插值多项式的次数
- 当插值多项式的次数升高时, 会带来不同程度的数值震荡, 即发生所谓的**龙格(Runge)现象:**

高次插值多项式的龙格现象

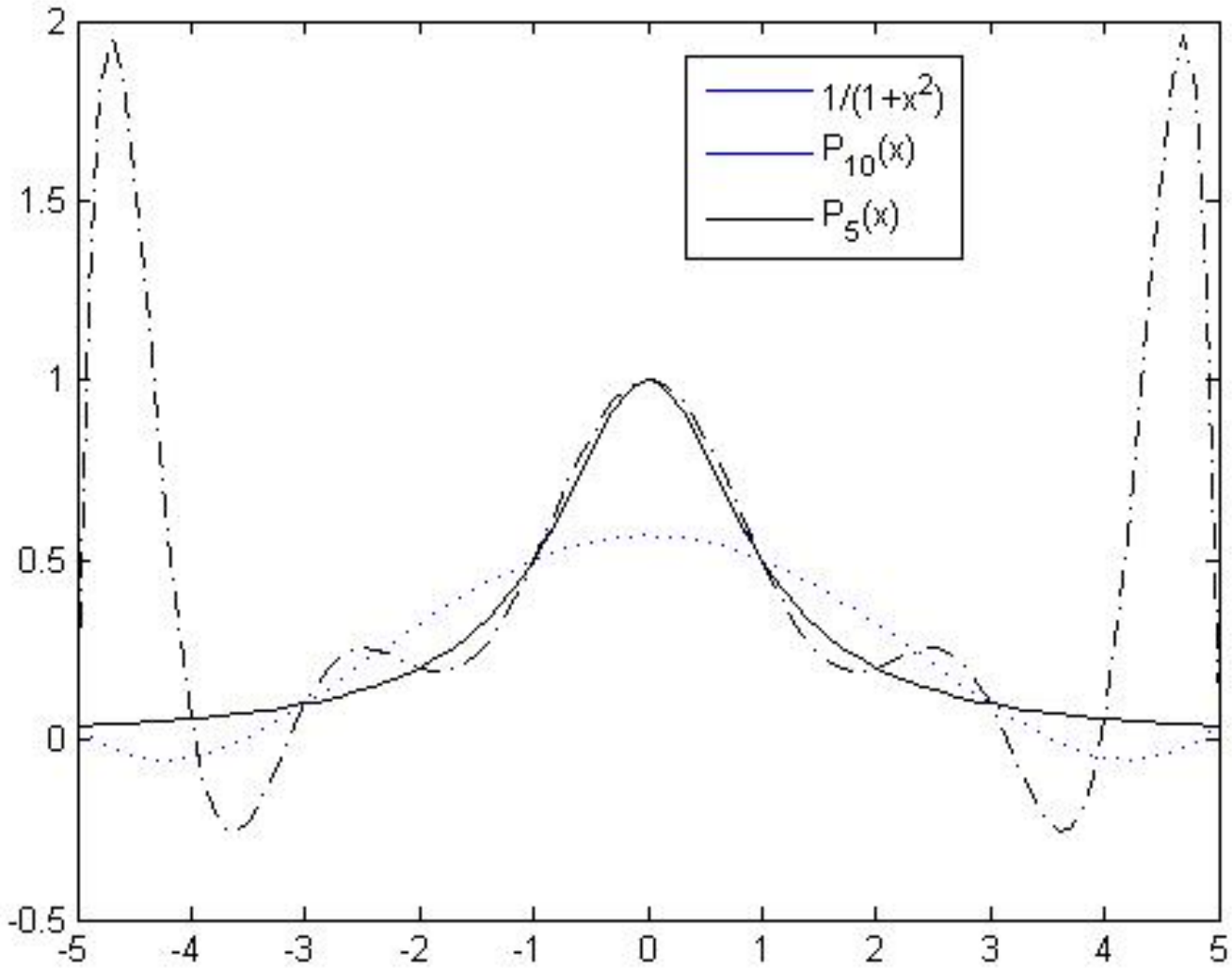
示例：

- 在区间 [-5,5]上, 生成 11 个等距插值节点, $x_i, i = 0, 1, \cdots, 10.$
- 在相应插值节点上计算函数

$$y(x) = 1/(1 + x^2)$$

  的函数值作为观测值, $y(x_i), i = 0, 1, 2, \cdots, 10.$
- 利用这 11 个数据点, 可以生成一个10 次拉格朗日插值多项式 $P_{10}(x)$.
- 类似地, 生成一个 5 次插值多项式 $P_5(x)$, 作图比较插值函数 $P_{10}(x), P_5(x)$ 和原函数 $f(x)$, 观察曲线拟合效果.

龙格震荡现象示意图



- Numerical Oscillation: the **oscillatory nature of high-degree polynomials** and the property that a **fluctuation over a small portion of the interval** can induce large fluctuations over the entire range restricts their use.
- To reduce numerical oscillation, one method is to **divide the interval into a collection of subintervals** and construct a (generally) different approximating polynomial on each subinterval. Approximation by functions of this type is called **piecewise polynomial approximation**.
- **The simplest piecewise polynomial approximation is piecewise linear interpolation, which consists of joining a set of data points**

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \cdots, (x_n, f(x_n))\}$$

by a series of straight line segments.

- Numerical Oscillation: the **oscillatory nature of high-degree polynomials** and the property that a **fluctuation over a small portion of the interval** can induce large fluctuations over the entire range restricts their use.
- To reduce numerical oscillation, one method is to **divide the interval into a collection of subintervals** and construct a (generally) different approximating polynomial on each subinterval. Approximation by functions of this type is called **piecewise polynomial approximation**.
- **The simplest piecewise polynomial approximation is piecewise linear interpolation, which consists of joining a set of data points**

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \cdots, (x_n, f(x_n))\}$$

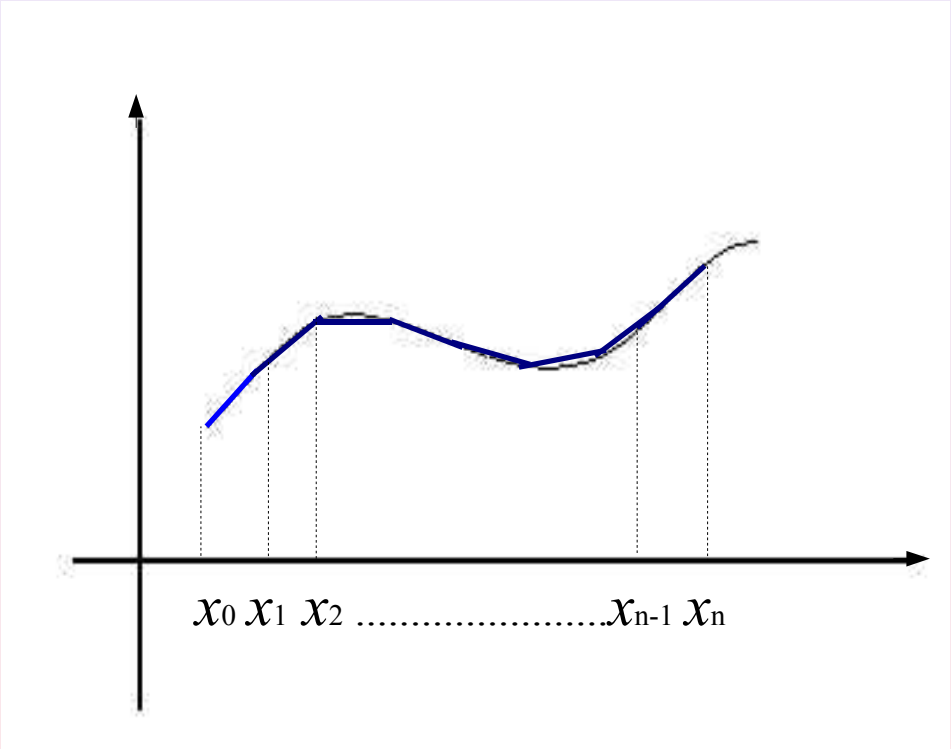
by a series of straight line segments.

- Numerical Oscillation: the **oscillatory nature of high-degree polynomials** and the property that a **fluctuation over a small portion of the interval** can induce large fluctuations over the entire range restricts their use.
- To reduce numerical oscillation, one method is to **divide the interval into a collection of subintervals** and construct a (generally) different approximating polynomial on each subinterval. Approximation by functions of this type is called **piecewise polynomial approximation**.
- **The simplest piecewise polynomial approximation is piecewise linear interpolation, which consists of joining a set of data points**

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \cdots, (x_n, f(x_n))\}$$

by a series of straight line segments.

- To find a linear polynomial $P(x)$ which is a linear combination of linear polynomial for each subinterval. Such that
  1. $P(x_i) = f(x_i)$, for each $i = x_0, x_1, \cdots, x_n$;
  2. $P(x)$ is linear polynomial in each subinterval $[x_{i-1}, x_i], i = x_0, x_1, \cdots, x_n$.
- See figure 3-3-1:


x_0 x_1 x_2 ..... x_{n-1} x_n

3.5.1 Piecewise Interpolation Linear Polynomial Approximation

- First construct base functions $l_i(x)$, which is linear for each subinterval $[x_{i-1}, x_i]$, $i = 1, 2, \cdots, n$, such that $l_i(x_j) = 0$ when $j \neq i$, and $l_i(x_i) = 1$:

$$l_0(x) = \begin{cases} \frac{x-x_1}{x_0-x_1}, & x_0 \leq x \leq x_1; \\ 0, & x_1 < x \leq x_n. \end{cases}$$

$$l_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}}, & x_{i-1} \leq x \leq x_i; \\ \frac{x-x_{i+1}}{x_i-x_{i+1}}, & x_i < x \leq x_{i+1}; \\ 0, & \texttt{others}. \end{cases}$$

for each inner nodes $i = x_1, x_2, \cdots, x_{n-1}$, and

$$l_n(x) = \begin{cases} \frac{x-x_{n-1}}{x_n-x_{n-1}}, & x_{n-1} \leq x \leq x_n; \\ 0, & x_0 \leq x < x_{n-1}. \end{cases}$$

- Thus the Piecewise Interpolation Linear Polynomial Approximation can be written as

$$\begin{aligned} P(x) &= f(x_0)l_0(x) + f(x_1)l_1(x) + \cdots + f(x_n)l_n(x) \\ &= \sum_{i=0}^{n} l_i(x)f(x_i). \end{aligned}$$

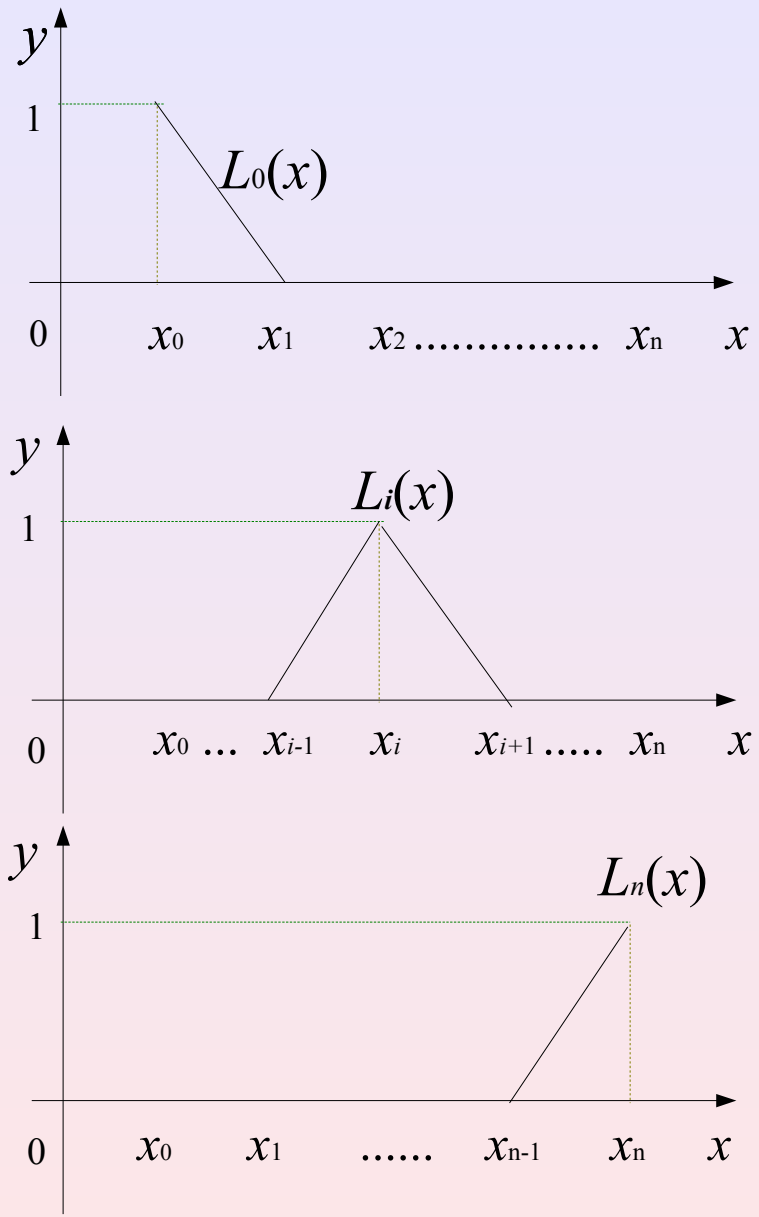

y
1
$L_0(x)$
0
x_0 x_1 x_2 ..... x_n x
y
1
$L_i(x)$
0
x_0 ... x_{i-1} x_i x_{i+1} ..... x_n x
y
1
$L_n(x)$
0
x_0 x_1 ..... x_{n-1} x_n x

Fig 3.5.2

THEOREM 3.10

- If $f \in C^1[a, b]$, and f'' exists on $[a, b]$
- $x_0, x_1, \cdots, x_n \in [a, b]$ are distinct numbers
- $P(x)$ is the Piecewise Interpolation Linear Polynomial Approximation defined by

$$P(x) = \sum_{i=0}^{n} l_i(x) f(x_i).$$

- Then we have the following error bounds

$$|f(x) - P(x)| \leq \frac{h^2}{8} M$$

where $h = \max_{0 \leq i \leq n-1} |x_{i+1} - x_i|$, $M = \max_{a \leq x \leq b} |f''(x)|$.■

NOTES:

- A **disadvantage of linear function approximation** is that there is **no assurance of differentiability** at each of the endpoints of the subintervals.
- In geometrical means, the interpolating function is not "smooth" at these points.
- It is clear from physical conditions that such a smoothness condition is required and that the approximating function must be continuously differentiable.

3.5.2 Piecewise Hermite Interpolation Polynomial Approximation

- An alternative procedure is to use a **piecewise polynomial of Hermite type**.
- For example, if the values of the function f and of f' are known at each of the points

$$x_0 < x_1 < \cdots < x_n.$$

- A Hermite polynomial of degree three can be used on each of the subintervals

$$[x_0, x_1], [x_1, x_2], \cdots, [x_{n-1}, x_n]$$

to obtain a function that is continuously differentiable on the interval $[x_0, x_n]$.

3.5.2 Piecewise Hermite Interpolation Polynomial Approximation

- If $f \in C^1[a, b]$ and $x_0, x_1, \cdots, x_n \in [a, b]$ are distinct
- The piecewise cubic Hermite polynomial at each subintervals $[x_{i-1}, x_i]$ agreeing with f and f' at endpoints $x_0, x_1, \cdots, x_n$ can be defined as

$$P_3(x) = \sum_{i=0}^{n} H_i(x) f(x_i) + \sum_{i=0}^{n} \hat{H}_i(x) f'(x_i).$$

where $H_i(x), \hat{H}_i(x), i = 0, 1, \cdots, n$ are base functions

$$H_0(x) = \begin{cases} \left(1 + 2\frac{x-x_0}{x_1-x_0}\right)\left(\frac{x-x_1}{x_0-x_1}\right)^2, & x_0 \le x \le x_1 \\ 0, & x_1 < x \le x_n. \end{cases}$$

$$H_i(x) = \begin{cases} \left(1 + 2\frac{x-x_i}{x_{i-1}-x_i}\right)\left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2, & x_{i-1} \le x \le x_i \\ \left(1 + 2\frac{x-x_i}{x_{i+1}-x_i}\right)\left(\frac{x-x_{i+1}}{x_i-x_{i+1}}\right)^2, & x_i < x \le x_{i+1} \\ 0, & \texttt{others}. \end{cases}$$

$$H_n(x) = \begin{cases} \left(1 + 2\frac{x-x_n}{x_{n-1}-x_n}\right)\left(\frac{x-x_{n-1}}{x_n-x_{n-1}}\right)^2, & x_{n-1} \le x \le x_n \\ 0, & x_0 \le x < x_{n-1}. \end{cases}$$

$$\hat{H}_0(x) = \begin{cases} (x - x_0)\left(\frac{x-x_1}{x_0-x_1}\right)^2, & x_0 \le x \le x_1 \\ 0, & x_1 < x \le x_n. \end{cases}$$

$$\hat{H}_i(x) = \begin{cases} (x - x_i)\left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2, & x_{i-1} \le x \le x_i \\ (x - x_i)\left(\frac{x-x_{i+1}}{x_i-x_{i+1}}\right)^2, & x_i < x \le x_{i+1} \\ 0, & \texttt{others}. \end{cases}$$

$$\hat{H}_n(x) = \begin{cases} (x - x_n)\left(\frac{x-x_{n-1}}{x_n-x_{n-1}}\right)^2, & x_{n-1} \le x \le x_n \\ 0, & x_0 \le x < x_{n-1}. \end{cases}$$

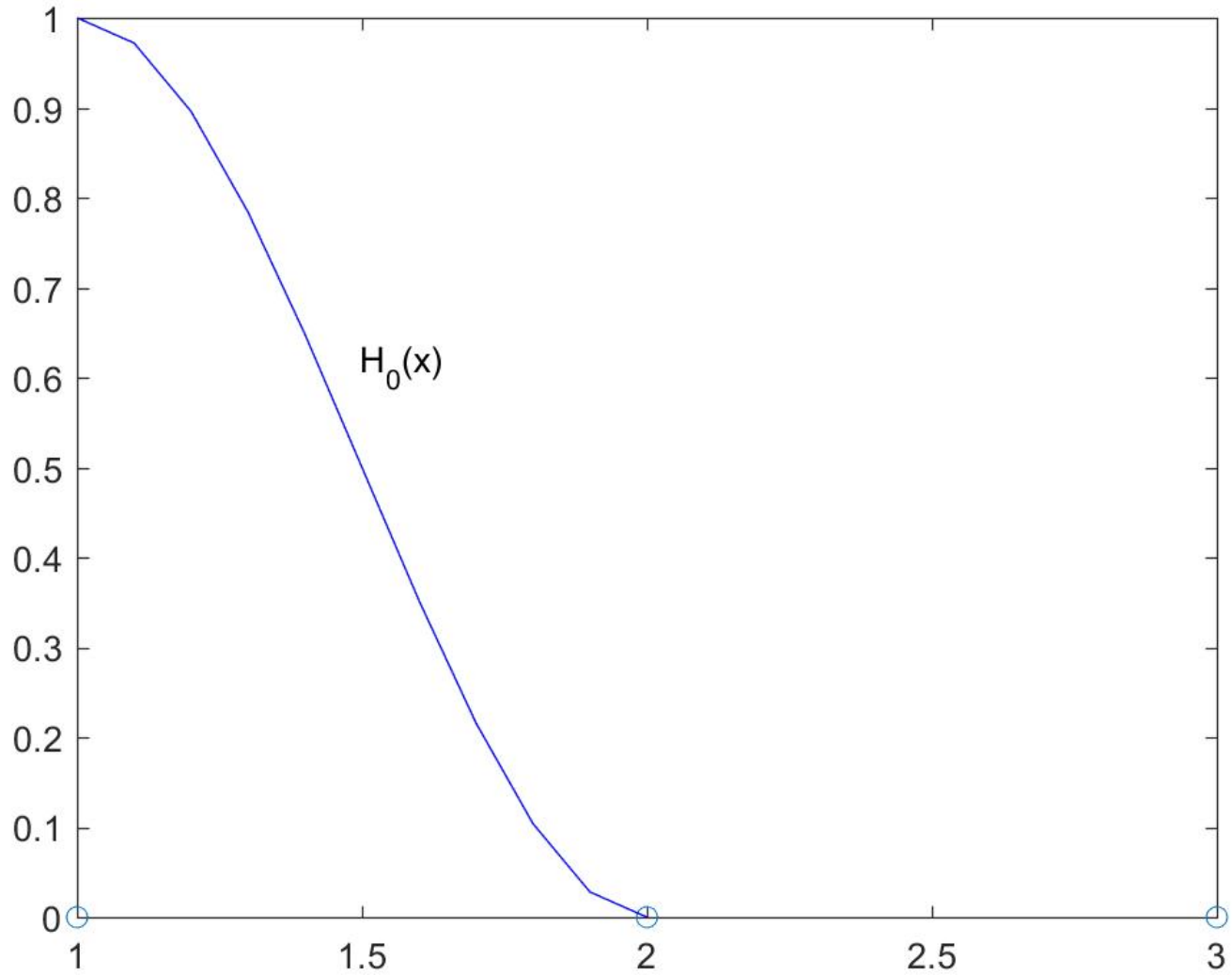


Figure: 3-4. (a) 在左边界点$x_0 = 1$ 处的插值基函数$H_0(x)$

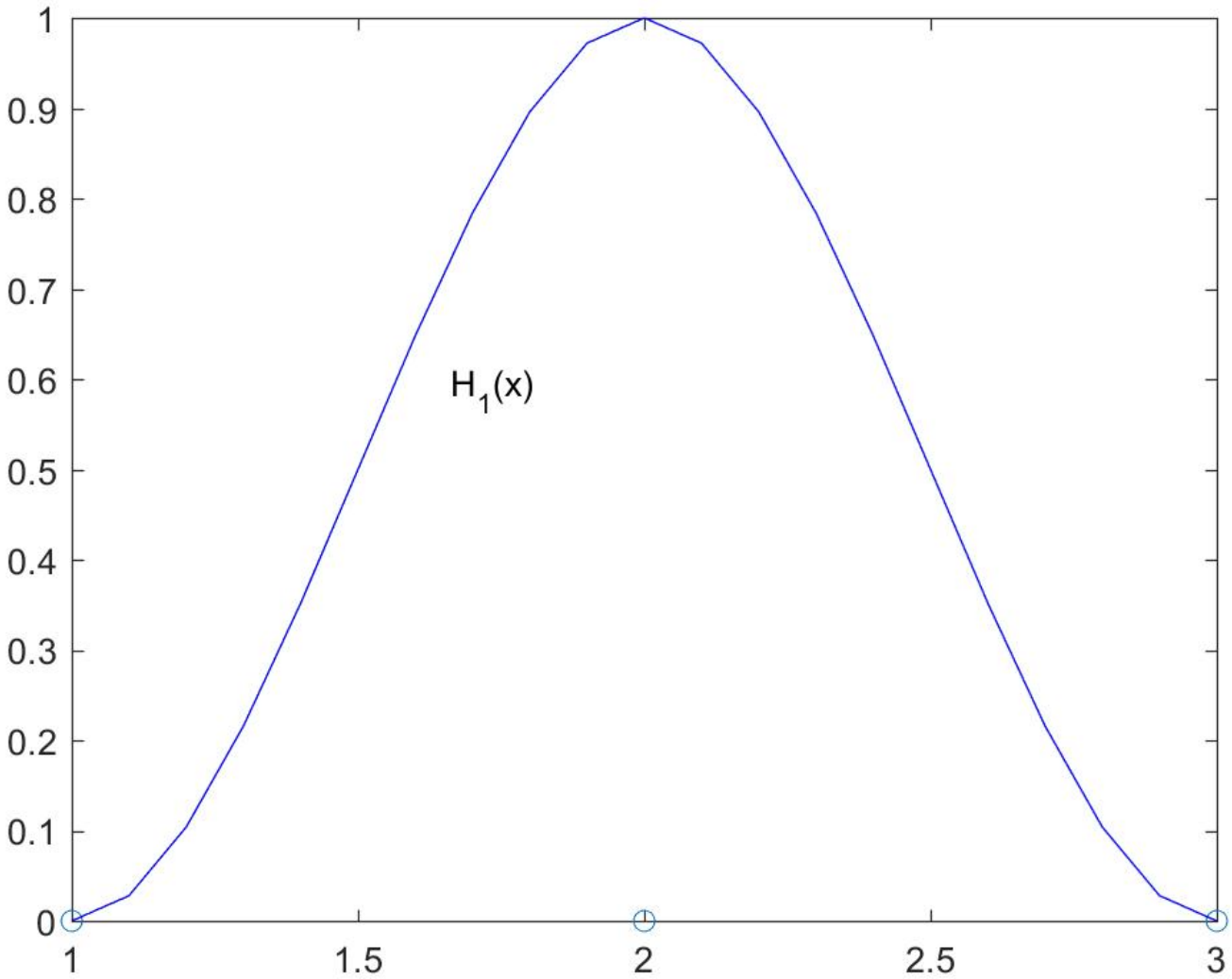


Figure: 3-4. (b) 在内点$x_1 = 2$ 处的插值基函数$H_1(x)$

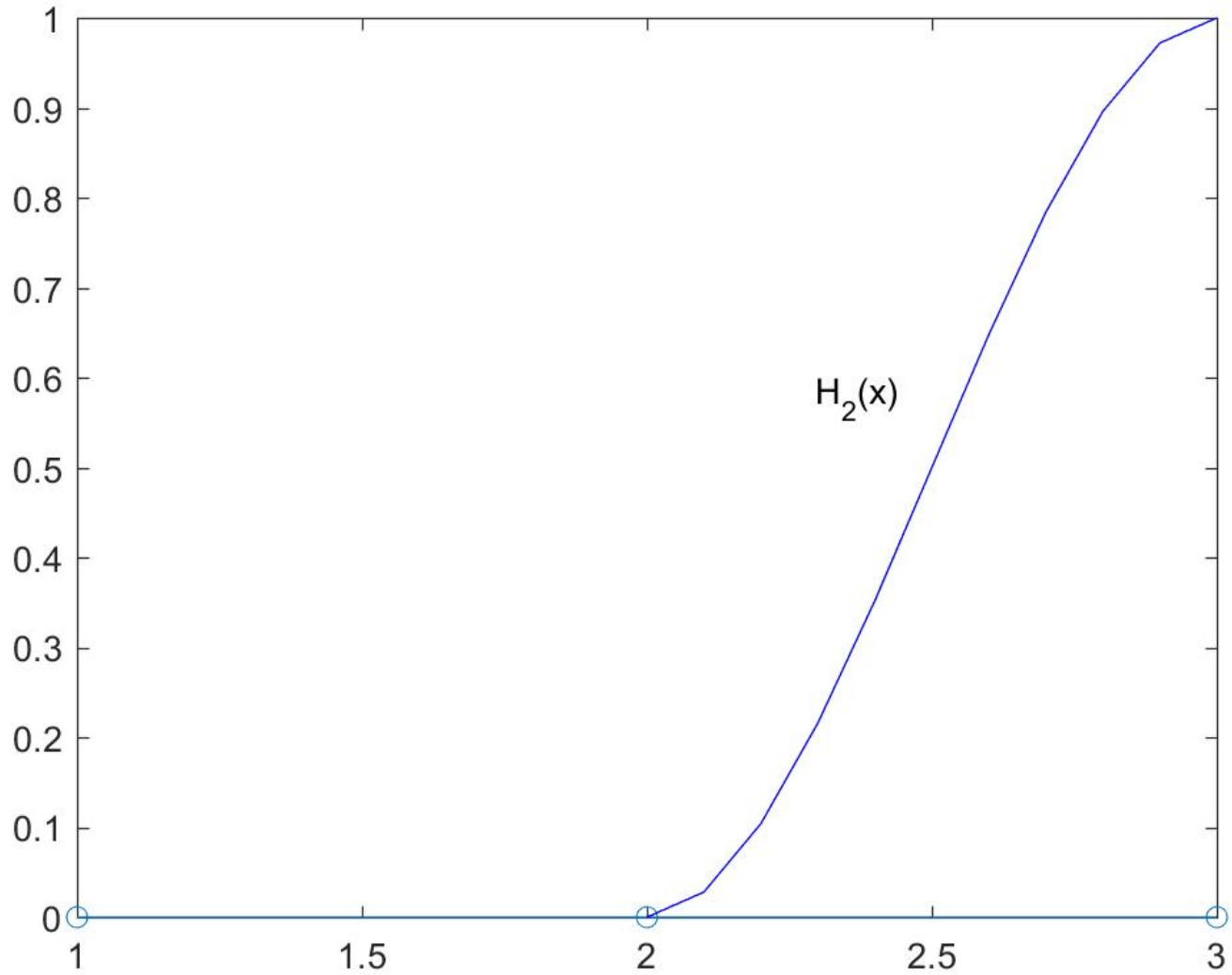


Figure: 3-4. (c) 在右边界点$x_2 = 3$ 处的插值基函数$H_2(x)$

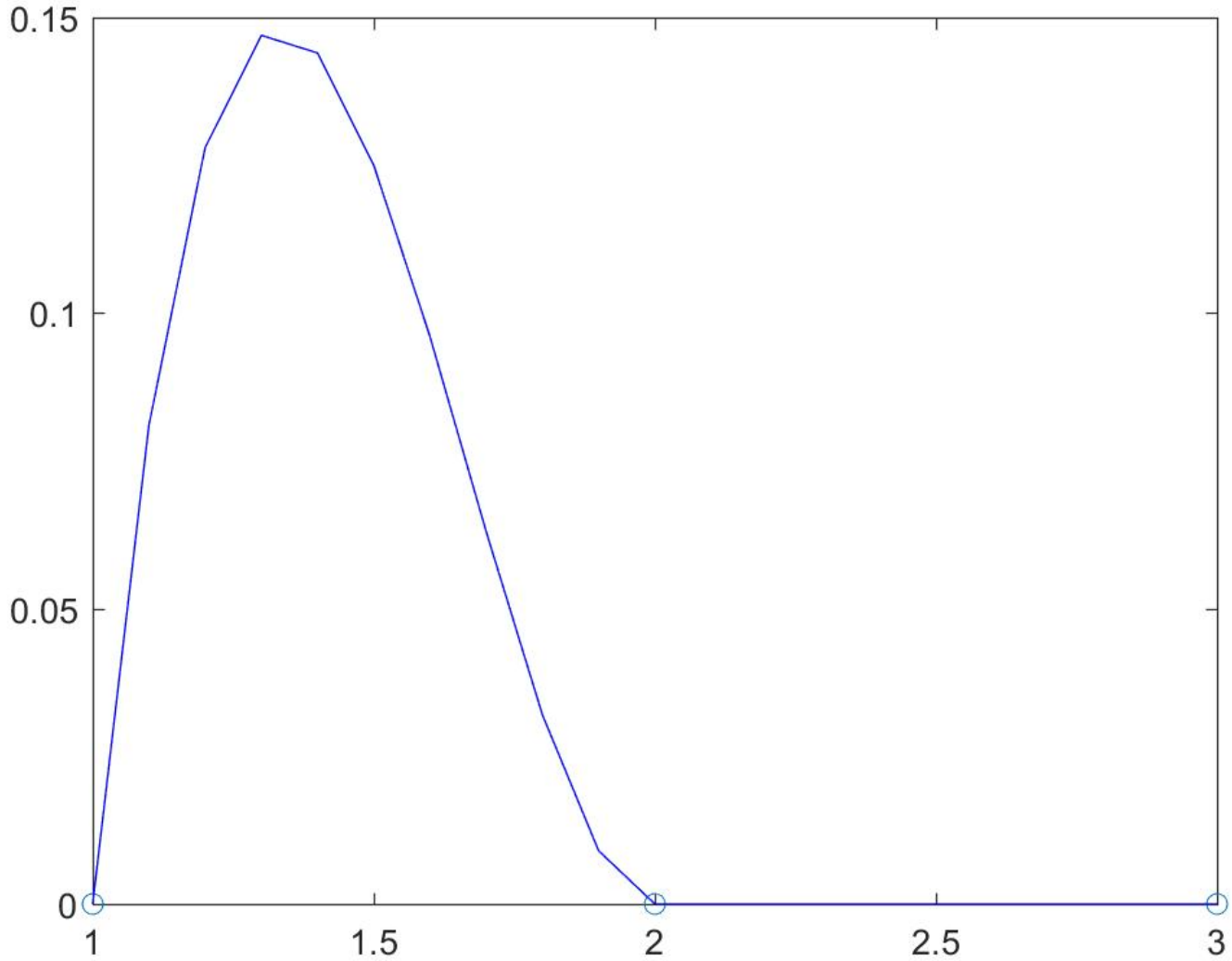


Figure: 3-5. (a) 在左边界点$x_0 = 1$ 处，关于导数的插值基函数$\hat{H}_0(x)$

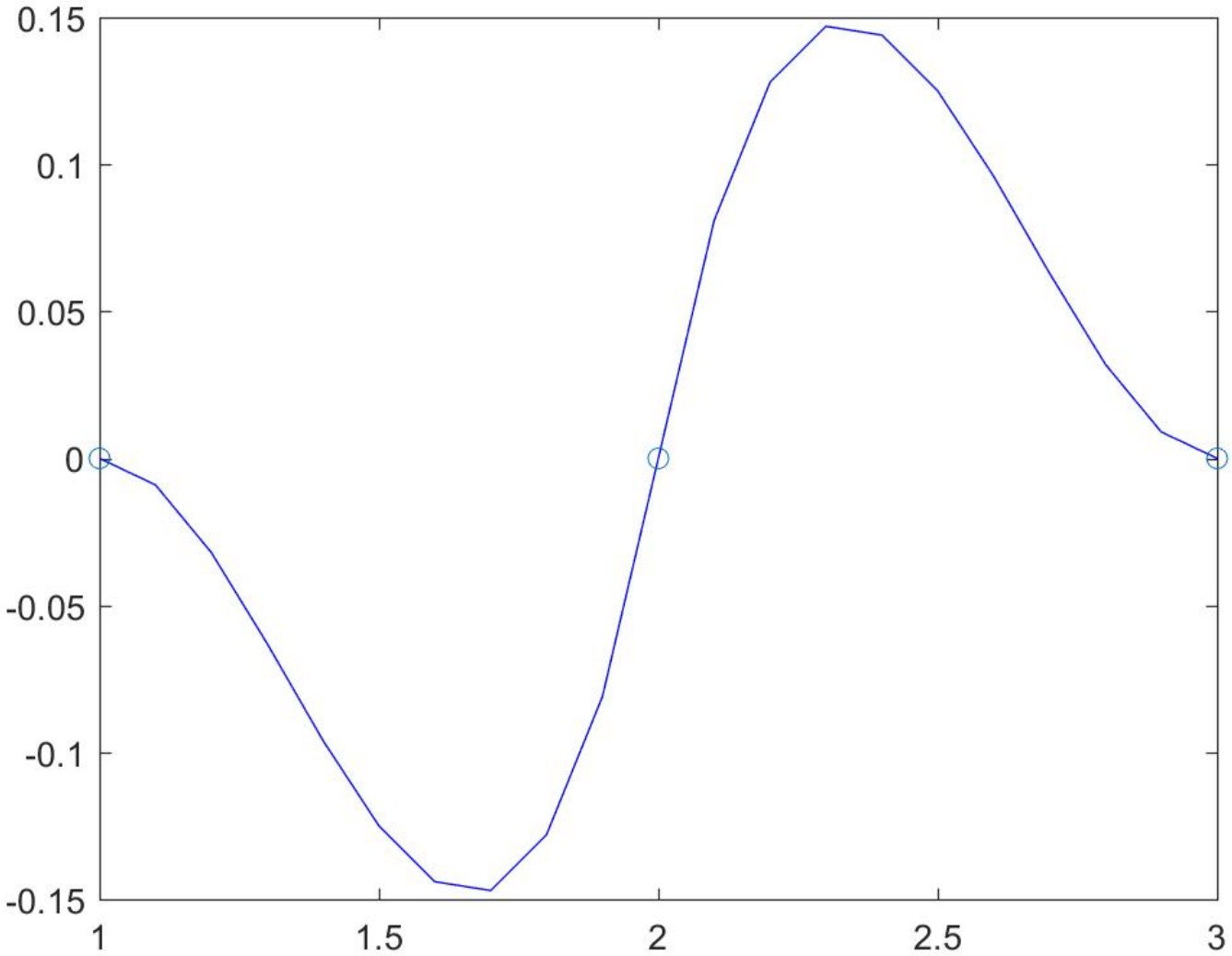


Figure: 3-5. (b) 在内点$x_1 = 2$ 处，关于导数的插值基函数$\hat{H}_1(x)$

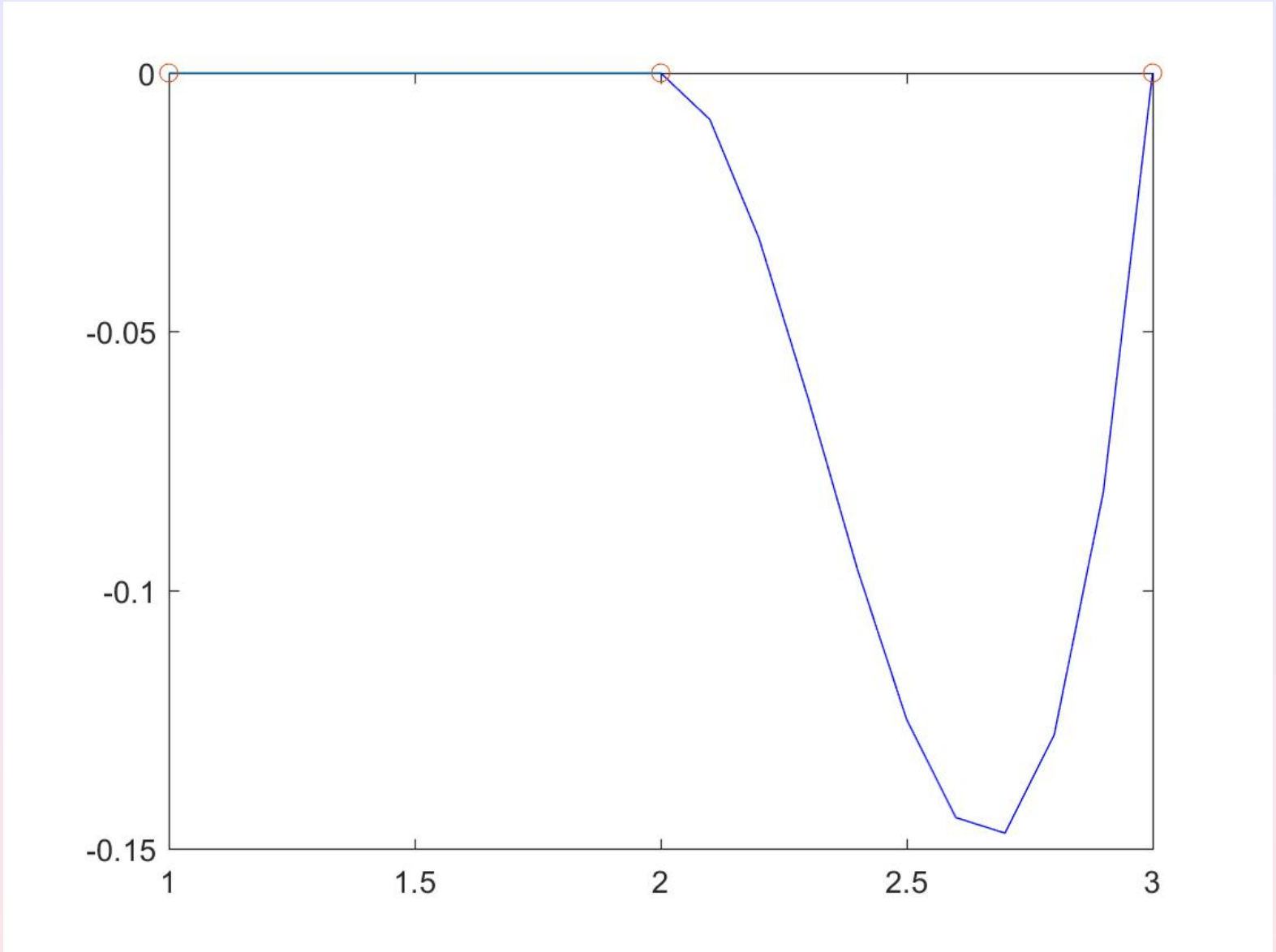


Figure: 3-5. (c) 在右边界点$x_2 = 3$ 处，关于导数的插值基函数$\hat{H}_2(x)$

关于分段插值的几点注释

- 高次插值多项式的光滑性高, 但震荡性也强
- 若插值节点相对比较密集时, 采用低次插值多项式也能得到很好的近似.
- **分段插值的基本思想：**
  1. 把插值区间划分割成若干个小区间或子区间（子区间的划分可以是任意的）.
  2. 在每一小区间上用低次多项式进行插值，理论上各个区间上插值多项式的次数的选取也可以视需要而定.
  3. 在整个插值区间上就得到一个分段插值函数

关于分段插值的几点注释

- 高次插值多项式的光滑性高, 但震荡性也强
- 若插值节点相对比较密集时, 采用低次插值多项式也能得到很好的近似.
- 分段插值的基本思想:
  1. 把插值区间划分割成若干个小区间或子区间（子区间的划分可以是任意的）.
  2. 在每一小区间上用低次多项式进行插值，理论上各个区间上插值多项式的次数的选取也可以视需要而定.
  3. 在整个插值区间上就得到一个分段插值函数

关于分段插值的几点注释

- 高次插值多项式的光滑性高，但震荡性也强
- 若插值节点相对比较密集时，采用低次插值多项式也能得到很好的近似.
- **分段插值的基本思想：**
  1. 把插值区间划分割成若干个小区间或子区间（子区间的划分可以是任意的）.
  2. 在每一小区间上用低次多项式进行插值，理论上各个区间上插值多项式的次数的选取也可以视需要而定.
  3. 在整个插值区间上就得到一个分段插值函数

备注

- 分段插值法通常有较好的收敛性和稳定性, 算法简单.
- 分段插值函数不如整体拉格朗日插值多项式光滑(通常在子区间的端点处不可导).
- 为了保证在区间端点(或部分端点)的光滑性（保形）,可增加相应端点的导数条件.
- 常用的分段插值方法有:
  - 局部光滑:分段线性插值、分段二次插值、分段三次插值等.
  - 样条插值

备注

- 分段插值法通常有较好的收敛性和稳定性, 算法简单.
- 分段插值函数不如整体拉格朗日插值多项式光滑(通常在子区间的端点处不可导).
- 为了保证在区间端点(或部分端点)的光滑性（保形）,可增加相应端点的导数条件.
- 常用的分段插值方法有:
  - 局部光滑:分段线性插值、分段二次插值、分段三次插值等.
  - 样条插值

备注

- 分段插值法通常有较好的收敛性和稳定性, 算法简单.
- 分段插值函数不如整体拉格朗日插值多项式光滑(通常在子区间的端点处不可导).
- 为了保证在区间端点(或部分端点)的光滑性（保形）,可增加相应端点的导数条件.
- 常用的分段插值方法有:
  - 局部光滑:分段线性插值、分段二次插值、分段三次插值等.
  - 样条插值

备注

- 分段插值法通常有较好的收敛性和稳定性, 算法简单.
- 分段插值函数不如整体拉格朗日插值多项式光滑(通常在子区间的端点处不可导).
- 为了保证在区间端点(或部分端点)的光滑性（保形）,可增加相应端点的导数条件.
- 常用的分段插值方法有:
  - 局部光滑:分段线性插值、分段二次插值、分段三次插值等.
  - 样条插值

DEFINITION 3.11

Cubic Spline Given a function f defined on $[a, b]$ and a set of nodes

$$a = x_0 < x_1 < \cdots < x_n = b,$$

a **cubic spline interpolation** $S(x)$ for $f(x)$ is a function that satisfies the following conditions:

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

b. $S(x_j) = f(x_j)$ for each $j = 0, 1, \cdots, n$;

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

Cubic Spline

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

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

f. One of the following set of boundary conditions is satisfied:

I $S''(x_0) = S''(x_n) = 0$, —free or natural boundary;

II $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$. —clamped boundary–固定边界条件

$$S_{j-1}(x_j) = S_j(x_j) = f(x_j)$$
$$S'_{j-1}(x_j) = S'_j(x_j)$$
$$S''_{j-1}(x_j) = S''_j(x_j)$$

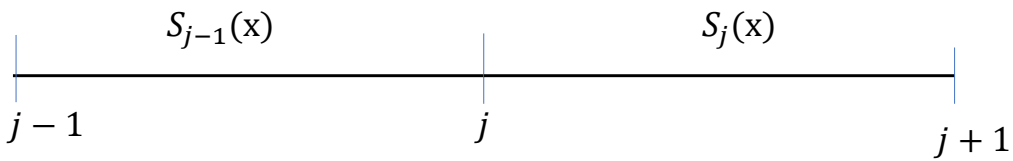


Fig.3-1 样条函数内点插值条件

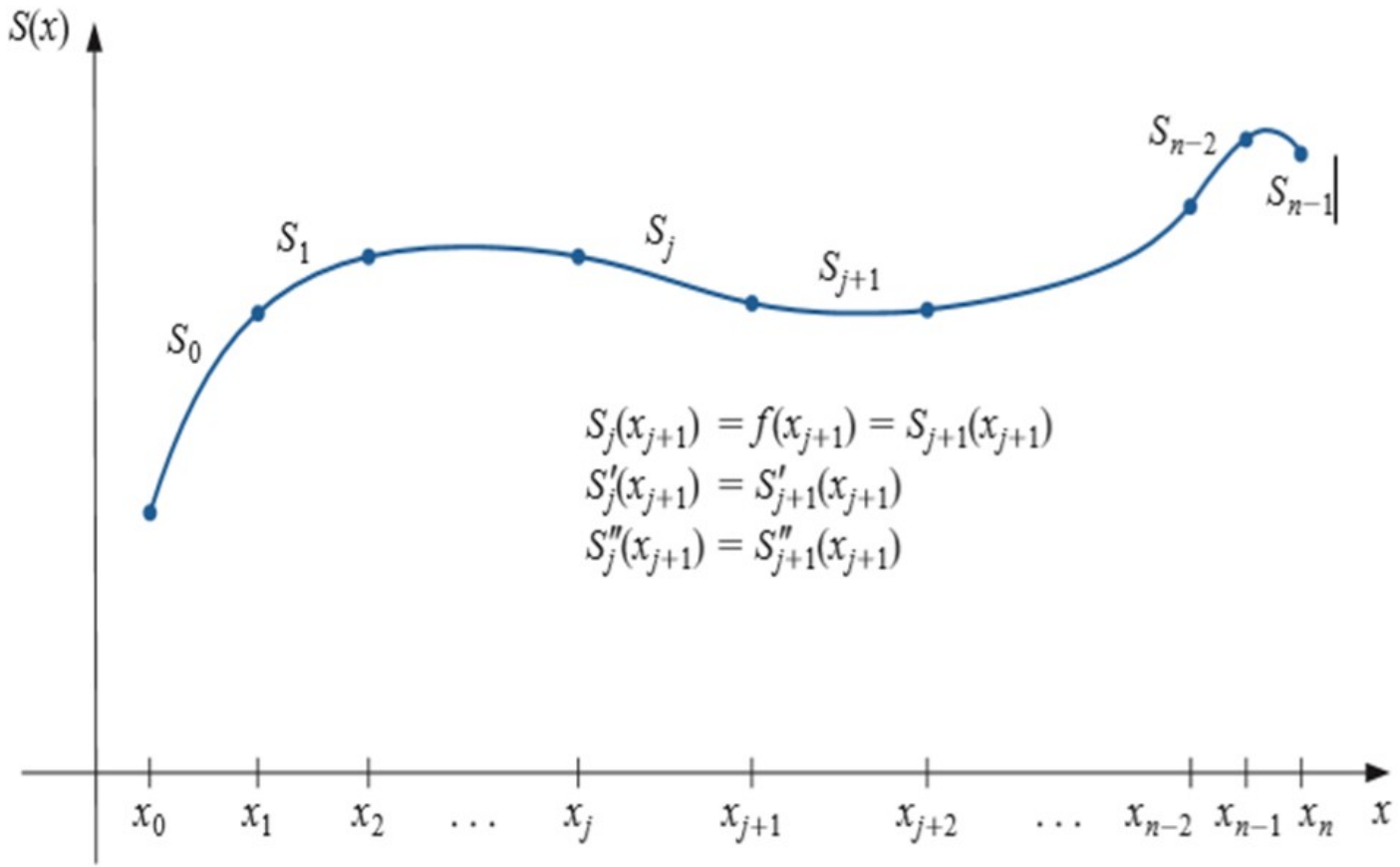


Figure: 3-2. 三次样条示意图

Construction of Cubic Spline Interpolant

- To construct the cubic spline interpolant for a given function f
- Let the cubic polynomials:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

on the subinterval

$$[x_j, x_{j+1}],\ j = 0, 1, \cdots, n-1.$$

- Then the left problem is to determine the coefficients

$$a_j, b_j, c_j, d_j . j = 0, 1, \cdots, n-1.$$

Clearly, by the condition (b), we can get

$$S_j(x_j) = a_j = f(x_j), j = 0, 1, \cdots, n$$

and applying condition (c) we obtain

$$\begin{aligned} a_{j+1} &= S_{j+1}(x_{j+1}) = S_j(x_{j+1}) \\ &= a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 \\ &\quad + d_j(x_{j+1} - x_j)^3 \end{aligned}$$

for each $j = 0, 1, \cdots, n - 2$.

- Notation: Let $h_j = x_{j+1} - x_j$, for each $j = 0, 1, \cdots, n-1$.
- If we also define $a_n = f(x_n)$, then the equation

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \qquad (1)$$

holds for each $j = 0, 1, \cdots, n-1$.

- In a similar manner, define $b_n = S'(x_n)$ and observe that

$$S_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$$

implies $S'(x_j) = b_j$ for each $j = 0, 1, \cdots, n-1$.

- Applying condition (d) gives

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2, \tag{2}$$

for each $j = 0, 1, \cdots, n-1$.

- Another relation between the coefficients of the S_j is obtained by defining $c_n = S''(x_n)/2$ and applying condition (e).
- In this case,

$$c_{j+1} = c_j + 3d_j h_j, \tag{3}$$

for each $j = 0, 1, \cdots, n-1$.

Solving for d_j by (3) , and substituting this value into (2) and (1) gives the new equations

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) \quad (4)$$

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}), \quad (5)$$

for each $j = 0, 1, \cdots, n-1$.

The final relationship involving the coefficients is obtained by solving the appropriate equation in the form of equation (4), first for b_j,

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}), \qquad (6)$$

and then, with a reduction of the index, for b_{j-1}. This gives:

$$b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j),$$

Substituting these values into the equation (5), with the index reduced by 1, gives the linear system of equations

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

for each $j = 1, \cdots, n-1$.

This system involves only $\{c_j\}_{j=0}^{n}$ as unknowns since the values of $\{h_j\}_{j=0}^{n-1}$ and $\{a_j\}_{j=0}^{n}$ are given by the spacing of the nodes $\{x_j\}_{j=0}^{n}$ and the values of f at the nodes.

Note that once the values of $\{c_j\}_{j=0}^{n}$ are known, it is a simple matter to find the remainder of the constants $\{b_j\}_{j=0}^{n-1}$ from (6) and $\{d_j\}_{j=0}^{n-1}$ from (3) and to construct the cubic polynomials $\{S_j(x)\}_{j=0}^{n-1}$.

THEOREM 3.12

If f is defined at $a = x_0 < x_1 < \cdots < x_n = b$, then f has a unique natural spline interpolant on the nodes $x_0, x_1, \cdots, x_n$; that is, a spline interpolant that satisfies the boundary conditions $S''(a) = 0$ and $S''(b) = 0$.■

Proof:

The boundary conditions in this case imply that $c_n = S''(x_n) = 0$ and that

$$0 = S''(x_0) = 2c_0 + 6d_0(x_0 - x_0);$$

so $c_0 = 0$.

The two equations $c_0 = 0$ and $c_n = 0$ together with the equations in (3.22) produce a linear system described by the vector equation $A\mathbf{x} = \mathbf{b}$, where A is the $(n+1) \times (n+1)$ matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & \ddots & \vdots \\ 0 & h_1 & 2(h_1+h_2) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \dots & \dots & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}.$$

The matrix A is strictly diagonally dominant, so the linear system has a unique solution for $c_0, c_1, \cdots, c_n$.■■■

Natural Cubic Spline ALGORITHM

To construct the cubic spline interpolant $S(x)$ for the function $f(x)$ defined at the distinct numbers $x_0, x_1, \cdots, x_n$ satisfying $S''(x_0) = S''(x_n) = 0$.

Input: $n; x_0, x_1, \cdots, x_n;$

$$a_0 = f(x_0), a_1 = f(x_1), \cdots, a_n = f(x_n);$$

Output: $a_j, b_j, c_j, d_j, \ j = 0, 1, 2, \cdots, n-1.$

Note that:

$$S(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3,$$

for all $x \in [x_j, x_{j+1}]$.

STEP 1 for $i = 0, 1, \cdots, n-1$ set $h_j = x_{j+1} - x_j$.

STEP 2 for $i = 1, 2, \cdots, n-1$ set

$$\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}).$$

STEP 3 Forming the coefficient matrix $\mathbf{A}$, and solving linear system $\mathbf{Ac} = \alpha$.

STEP 5 for $j = n-1, n-2, \cdots, 1, 0$ set

$$b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3;$$

$$d_j = (c_{j+1} - c_j)/(3h_j).$$

STEP 6 OUTPUT a_j, b_j, c_j, d_j for $j = 0, 1, \cdots, n-1$. STOP.

THEOREM 3.13

If f is defined at $a = x_0 < x_1 < \cdots < x_n = b$ and differentiable at a and b, then f has a unique clamped spline interpolant on the nodes $x_0, x_1, \cdots, x_n$, that is, a spline interpolant that satisfies the boundary conditions $S'(a) = f'(a)$ and $S'(b) = f'(b)$.■

PROOF:

Since $S'(a) = f'(a) = S'(x_0) = b_0$, Equation (6), with $j = 0$ implies

$$S_0'(x_0) = S_0'(a) = b_0 = \frac{a_1 - a_0}{h_0} - \frac{h_0(2c_0 + c_1)}{3} = f'(a).$$

$\Rightarrow$

$$2h_0 c_0 + h_0 c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a).$$

Similarly, By $S'(b) = f'(b)$, using the formula (5) and (6) with $j = n - 1$, implies

$$\begin{aligned} f'(b) = b_n &= b_{n-1} + h_{n-1}(c_{n-1} + c_n) \\ &= \frac{a_n - a_{n-1}}{h_{n-1}} - \frac{h_{n-1}}{3}(2c_{n-1} + c_n) \\ &\quad + h_{n-1}(c_{n-1} + c_n) \\ &= \frac{a_n - a_{n-1}}{h_{n-1}} + \frac{h_{n-1}}{3}(c_{n-1} + 2c_n) \end{aligned}$$

$$\Rightarrow$$

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n$$

$$= 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}).$$

Combine

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1}$$

$$= \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

with above two formula, forming linear system $\mathbf{Ax} = \mathbf{b}$.

where

$$\mathbf{A} = \begin{pmatrix} 2h_0 & h_0 & 0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & \ddots & \vdots \\ 0 & h_1 & 2(h_1+h_2) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} \frac{3}{h_0}(a_1 - a_0) - 3f'(a) \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{pmatrix}, \mathbf{x} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}.$$

The matrix $\mathbf{A}$ is strictly diagonally dominant, so the linear system has a unique solution for $c_0, c_1, \cdots, c_n$.■■■

THEOREM 3.14

- Let $f \in C^4[a, b]$ with

$$\max_{a \le x \le b} |f^{(4)}(x)| = M.$$

- If S is unique clamped cubic spline interpolant to f with respect to the nodes

$$a = x_0 < x_1 < \cdots < x_n = b.$$

- then

$$\max_{a \le x \le b} |f(x) - S(x)| \le \frac{5M}{384} \max_{0 \le j \le n-1} (x_{j+1} - x_j)^2. \blacksquare$$

Clamped Cubic Spline ALGORITHM

To construct the cubic spline interpolant $S(x)$ for the function $f(x)$ defined at the distinct numbers $x_0, x_1, \cdots, x_n$ satisfying $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$.

INPUT:

$$n; x_0, x_1, \cdots, x_n;$$
$$a_0 = f(x_0), a_1 = f(x_1), \cdots, a_n = f(x_n);$$
$$FPO = f'(x_0), FPV = f'(x_n).$$

OUTPUT: a_j, b_j, c_j, d_j for $j = 0, 1, 2, \cdots, n-1$.

Note that:

$$S(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3,$$

for any $x \in [x_j, x_{j+1}]$.

STEP 1 For $i = 0, 1, \cdots, n-1$ set $h_j = x_{j+1} - x_j$.

STEP 2 set

$$\alpha_0 = 3(a_1 - a_0)/h_0 - 3FTO;$$

$$\alpha_n = 3FTV - 3(a_n - a_{n-1})/h_{n-1}.$$

STEP 3 for $i = 1, 2, \cdots, n-1$ set

$$\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}).$$

STEP 4 Forming the coefficient matrix A, and solving linear system $\mathbf{Ac} = \alpha$.

STEP 5 for $j = n-1, n-2, \cdots, 1, 0$, set

$$b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3;$$

$$d_j = (c_{j+1} - c_j)/(3h_j).$$

STEP 6 OUTPUT $(a_j, b_j, c_j, d_j$ for $j = 0, 1, \cdots, n-1$. STOP.