Chapter 5 Interpolation and Polynomial Approximation

Fan Yang

Lan Zhou University of Technology,
Lanzhou, 730050, P.R. China
Outline

1. Interpolating Polynomials
   - Existence and Uniqueness of Interpolating Polynomials

2. Lagrange Interpolating
   - Lagrange Interpolating Polynomials
   - Lagrange Error Terms

3. Newton Interpolation
   - Divided Differences
   - Newton’s interpolating polynomial

4. Hermite Interpolation

5. Piecewise Polynomial Interpolation
   - Runge’s Phenomenon
   - Piecewise Linear Interpolation
   - Piecewise Cubic Hermite Interpolation

6. Cubic Spline Interpolation
   - Definition of Cubic Spline
   - The Construction of Cubic Spline

7. Least Squares Method
   - Least Squares Method
   - Least Squares Method in Inner Product
(Existence and Uniqueness.) If the points

\[(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))\]

are given, where \(x_0, x_1, \ldots, x_n\) are distinct, then there exists a unique polynomial \(p_n \in \mathbb{P}_n\) such that

\[p_n(x_i) = f(x_i), \text{ for } i = 0, 1, 2, \ldots, n.\]  \hspace{1cm} (5.1.1)
Suppose a polynomial,

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \in \mathbb{P}_n,$$

satisfies the interpolating conditions in (5.1.1), that is,

$$
\begin{align*}
    p_n(x_0) &= a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n = f(x_0), \\
    p_n(x_1) &= a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n = f(x_1), \\
    & \vdots \\
    p_n(x_n) &= a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n = f(x_n).
\end{align*}
$$

(5.1.2) can be viewed as a linear system of equations for
Proof

The determinant of $A$, $\det A$, which is a Vandermonde determinant, is not equal to zero since $x_0, x_1, \cdots, x_n$ are distinct.
Proof

The theory in Linear Algebra confirms that the system (5.1.2) has a unique solution $a_0, a_1, \ldots, a_n$, which corresponds to a unique polynomial $p_n(x)$ in $P_n$ satisfying the interpolating conditions in (5.1.1).
Theorem

(Lagrange Interpolating Polynomials.) If \( x_0, x_1, \cdots, x_n \) are \( n+1 \) distinct numbers and \( f \) is a function whose values are given at these numbers:

\[
f(x_i) = y_i, \text{ for } i = 0, 1, 2, \cdots, n, \tag{5.2.1}
\]

then the polynomial

\[
p_n(x) = y_0 l_0(x) + y_1 l_1(x) + \cdots + y_n l_n(x) \tag{5.2.2}
\]

satisfies the interpolating conditions.
\[ p_n(x_i) = y_i, \text{ for each } i = 0, 1, 2, \cdots, n, \quad (5.2.3) \]

where

\[ l_k(x) = \prod_{\substack{i = 0 \atop i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}, \text{ for } k = 0, 1, 2, \cdots, n. \quad (5.2.4) \]

The polynomial \( p_n(x) \) defined (5.2.2) is called the \textit{n}th \textit{Lagrange interpolating polynomial} for \( f(x) \), and \( l_0(x), l_1(x), \cdots, l_n(x) \) are called \textit{Lagrange coefficient polynomials} or \textit{Lagrange bases}. 

Fan Yang: yfggd114@163.com
Proof

If we construct polynomials $l_0(x), l_1(x), \ldots, l_n(x) \in \mathbb{P}_n$ with the property:

$$l_k(x_i) = \delta_{ki} = \begin{cases} 0, & \text{if } k \neq i, \\ 1, & \text{if } k = i, \end{cases} \quad \text{for each } i, \ k = 0, 1, \ldots, n,$$

(5.2.5)
then the polynomial

\[ p_n(x) = y_0 l_0(x) + y_1 l_1(x) + \cdots + y_n l_n(x) \]

satisfies the interpolating conditions

\[ p_n(x_i) = y_i, \text{ for each } i = 0, 1, 2, \cdots, n. \]

For each \( k \), (5.2.5) tells that \( l_k(x) \) has \( n \) zeroes \( x_0, \cdots, x_{k-1}, x_{k+1}, \cdots, x_n \). So, from the knowledge in Algebra, \( l_k(x) \) can be written in the form:
$l_k(x) = A_k(x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n), \ (5.2.6)$

where $A_k$ is a constant.

Substituting $x_k$ for $x$ in $(5.2.6)$ with $l_k(x_k) = 1$ gives

$$A_k(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n) = 1,$$

and then

$$A_k = \frac{1}{(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}.$$

Hence we get
Lagrange Interpolating Polynomials

Lagrange Interpolating Polynomials

Lagrange Error Terms

\[ l_k(x) = \frac{(x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)} \]

\[ = \prod_{i=0}^{n} \frac{x - x_i}{x_k - x_i} \]

**Example**

Find a Lagrange interpolating polynomial for \( f(x) \) using the data \( f(-1) = 1, f(1) = -1, f(2) = 2 \).

Fan Yang: yfggd114@163.com

Interpolation
Solution

If we denote

\[ x_0 = -1, \ x_1 = 1, \ x_2 = 2; \ y_0 = 1, \ y_1 = -1, \ y_2 = 2, \]

then (5.2.4) gives

\[
\begin{align*}
    l_0(x) &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x - 1)(x - 2)}{(-1 - 1)(-1 - 2)} = \frac{1}{6}(x^2 - 3x + 2), \\
    l_1(x) &= \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x + 1)(x - 2)}{(1 + 1)(1 - 2)} = -\frac{1}{2}(x^2 - x - 2),
\end{align*}
\]
and

\[ l_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(x + 1)(x - 1)}{(2 + 1)(2 - 1)} = \frac{1}{3}(x^2 - 1). \]

From (5.2.2) we get

\[ p_2(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x) = \frac{1}{3}(4x^2 - 3x - 4), \]

which is the Lagrange interpolating polynomial for \( f(x) \) we want to find.
Example

Find a Lagrange interpolating polynomial that passes through point A(−1, 1), B(0, 1), and C(2, 5).

Solution

If we denote

\[ x_0 = -1, \ x_1 = 0, \ x_2 = 2; \ y_0 = -1, \ y_1 = 1, \ y_2 = 5, \]
Solution

then (5.2.4) gives

\[ l_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x - 0)(x - 2)}{(-1 - 1)(-1 - 2)} = \frac{1}{3} (x^2 - 2x), \]
and

\[ l_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x + 1)(x - 2)}{(0 + 1)(0 - 2)} = -\frac{1}{2}(x^2 - x - 2), \quad l_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(x + 1)(x - 2)}{(0 + 1)(0 - 2)} = -\frac{1}{2}(x^2 - x - 2), \]

From (5.2.2) we get

\[ p_2(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x) = 2x + 1. \]
If we introduce a notation

$$\omega_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n),$$ \hspace{1cm} (5.2.7)

then the Lagrange coefficient polynomials $l_k(x)$ can be written in the following compact form

$$l_k(x) = \prod_{\substack{i = 0 \atop i \neq k}}^{n} \frac{x - x_i}{x_k - x_i} = \frac{\omega_n(x)}{(x - x_k)\omega'_n(x_k)} \text{ for } k = 0, 1, 2, \cdots, n.$$ \hspace{1cm} (5.2.8)
In fact, since

\[ \omega_n(x) = \prod_{i=0}^{n} (x - x_i) = (x - x_k) \prod_{i=0, i \neq k}^{n} (x - x_i), \quad k = 1, 2, \ldots, n. \]

So

\[ \omega'_n(x) = \prod_{i=0, i \neq k}^{n} (x - x_i) + (x - x_k) \frac{d}{dx} \left[ \prod_{i=0, i \neq k}^{n} (x - x_i) \right]. \]

Thus we obtain

\[ \omega'_n(x_k) = \prod_{i=0, i \neq k}^{n} (x_k - x_i), \quad k = 1, 2, \ldots, n. \]

we have
Therefore

\[ l_k(x) = \prod_{i=0}^{n} \frac{x - x_i}{x_k - x_i} = \frac{\omega_n(x)}{(x - x_k)\omega'_n(x_k)}. \]
Lagrange Error Terms

Theorem

(Lagrange Error Formula.) Suppose $x_0, x_1, \ldots, x_n$ are distinct numbers in the interval $[a, b]$ and $f \in C^{n+1}[a, b]$. Then, for each $x \in [a, b]$, there exist a number $\xi = \xi(x) \in (a, b)$ such that

$$f(x) = p_n(x) + r_n(x), \quad (5.2.9)$$
where

\[ r_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_n(x), \]  

(5.2.10)

\( r_n(x) \) is called the \textit{interpolating remainder term} associated with \( p_n(x) \), and \( p_n(x) \) is the interpolating polynomial given in (5.2.2).
Proof

Note first that if \( x = x_k \), for any \( k = 0, 1, \ldots, n \), then \( p_n(x_k) = f(x_k) \), and choosing \( \xi(x_k) \) arbitrarily in \((a, b)\) yields (5.2.9).

If \( x \neq x_k \), for all \( k = 0, 1, \ldots, n \), due to \( p_n(x_i) = y_i (i = 0, 1, \ldots, n) \), so \( x_i (i = 0, 1, \ldots, n) \) are all the zero of \( r_n(x) \), Thus we can assume

\[
r_n(x) = k(x)w_n(x),
\]

where \( k(x) \) is unknown.
\[\forall x \in [a, b] \text{ and } x \neq x_i, \text{ we define}\]

\[F(t) = f(t) - p_n(t) - k(x)w_n(t).\]  

(5.2.11)

Since \( f \in C^{n+1} [a, b] \), and \( p_n \in C^\infty [a, b] \), it follows that \( F \in C^{n+1} [a, b] \). For \( t = x_k, k = 0, 1, \cdots, n \), we have

\[F(x_k) = f(x_k) - p_n(x_k) - k(x)\omega_n(x_k) = 0,\]
\[ F(x) = f(x) - p_n(x) - k(x)\omega_n(x) = 0. \]

Thus, \( F \) has \( n + 2 \) distinct numbers zero points at the \( x, x_0, x_1, \ldots, x_n \). Applying Roll’s Theorem to \( F(t) \) on intervals
\[
[x_0, x], [x, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n],
\]
respectively, we know $F'(t) = 0$ has at least $n + 1$ zero points in $[a, b]$. So $F^{n+1}(t)$ has at least one zero.

$$0 = F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - k(x)(n + 1)! \quad (5.2.14)$$

So $k(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}$. 

Fan Yang: yfggd114@163.com
Corollary

Suppose $x_0, x_1, \cdots, x_n$ are distinct numbers in the interval $[a, b]$ and $f \in \mathbb{P}_n$. Then the Lagrange interpolating polynomial associated with $f(x)$ on $x_0, x_1, \cdots, x_n$ is identically $f(x)$ itself.
Proof

Suppose the Lagrange interpolating polynomial associated $f(x)$ on $x_0$, $x_1$, $\cdots$, $x_n$ is $p_n(x)$. Theorem 5.2.2 gives

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n), \quad (5.2.15)$$

where $\xi(x) \in (a, b)$. Since $f(x)$ is a polynomial of degree at most $n$, the $(n+1)$st derivative $f^{(n+1)}(x)$ is identically zero. So (5.2.15) becomes

$$p_n(x) = f(x).$$
Theorem

Suppose \( x_0, x_1, \cdots, x_n \) are distinct numbers in the interval \([a, b]\). Then the Lagrange coefficient polynomials \( l_0(x), l_1(x), \cdots, l_n(x) \) have the following properties:

(a) \( \sum_{k=0}^{n} l_k(x) \equiv 1; \)

(b) \( \sum_{k=0}^{n} x_k^m l_m(x) \equiv x^m, \) for \( m = 0, 1, \cdots, n; \)

(c) \( \sum_{k=0}^{n} (x_k - x)^m l_k(x) \equiv 0, \) for \( m = 1, 2, \cdots, n. \)
Proof

(a) Let \( f(x) \equiv 1 \), then

\[
y_k = f(x_k) = 1, \text{ for } k = 0, 1, \ldots, n.
\]

Theorem 5.2.1 implies that the Lagrange interpolating polynomial \( p_n(x) \) of \( f(x) \) on the nodes \( x_0, x_1, \ldots, x_n \) is
From Corollary 5.2.1 we know $p_n(x) \equiv f(x) \equiv 1$, so

$$\sum_{k=0}^{n} l_k(x) \equiv 1.$$

(b) Let $f(x) \equiv x^m$, then

$$y_k = f(x_k) = x_k^m, \text{ for } k = 0, 1, \cdots, n.$$
polynomial $p_n(x)$ of $f(x)$ on the nodes $x_0, x_1, \cdots, x_n$ is

$$p_n(x) = \sum_{k=0}^{n} y_k l_k(x) = \sum_{k=0}^{n} x_k^m l_k(x).$$

From Corollary 5.2.1 we know $p_n(x) \equiv f(x) = x^m$, so

$$\sum_{k=0}^{n} x_k^m l_m(x) \equiv x^m.$$

(c) For $m = 1, 2, \cdots, n,$
Interpolating Polynomials

Lagrange Interpolating Polynomials

Newton Interpolation

Hermite Interpolation

Piecewise Polynomial Interpolation

Cubic Spline Interpolation

Least Squares Method

Least Squares Approximation

Orthogonal Polynomials

Lagrange Interpolating Polynomials

Lagrange Error Terms

\[
\sum_{k=0}^{n} (x_k - x)^m l_k(x) = \sum_{k=0}^{n} \left[ \sum_{j=0}^{m} C_m^j x_k^j (-x)^{m-j} \right] l_k(x)
\]

\[
= \sum_{j=0}^{m} C_m^j (-x)^{m-j} \left[ \sum_{k=0}^{n} x_k^j l_k(x) \right]
\]
\[ C_m^j = \binom{m}{j} = \frac{m!}{j!(m-j)!} \]

Here \( C_m^j \) is the coefficient in the Lagrange polynomial.
Exercise

1. Given \( \sin\left(\frac{\pi}{6}\right) = 0.5000, \sin\left(\frac{\pi}{4}\right) = 0.7071, \sin\left(\frac{\pi}{3}\right) = 0.8660 \), using the 1 and 2 Lagrange interpolating polynomial, respectively, to approximate value of \( \sin\left(\frac{2\pi}{9}\right) \), moreover, compute the error estimate.

2. Suppose \( f(x) = x^4 \), using Lagrange Error Formula and \((-1, 1), (0, 0), (1, 1), (3, 9)\) to compute \( P_3(x) \), which is a 3 power polynomial.

3. Suppose \( f(x) = x^4 \), using Lagrange Error Formula and \((-1, 1), (0, 0), (1, 1), (3, 9), (4, 16)\) to compute \( P_4(x) \), which is a 4 power polynomial.
4. Find a Lagrange interpolating polynomial for $f(x)$ using the data 

(0, 1), (1, 2), (2, 3).

5. Find a Lagrange interpolating polynomial for $f(x)$ using the data 

$f(-1) = 3, \ f(1) = 0, \ f(2) = 5, \ f(4) = 4.$
Contents

1 Interpolating Polynomials
   • Existence and Uniqueness of Interpolating Polynomials

2 Lagrange Interpolating
   • Lagrange Interpolating Polynomials
   • Lagrange Error Terms

3 Newton Interpolation
   • Divided Differences
   • Newton’s interpolating polynomial

4 Hermite Interpolation

5 Piecewise Polynomial Interpolation
   • Runge’s Phenomenon
   • Piecewise Linear Interpolation
   • Piecewise Cubic Hermite Interpolation

6 Cubic Spline Interpolation

Fan Yang: yfggd114@163.com
Because the Lagrange interpolation formula lacks the recurrence relation between the high order interpolation and low interpolation. Each new increase node needs to be recalculated the Lagrange bases function, the old Lagrange bases function can not be used again. The high order interpolation can not make use of the result of low interpolation. In order to overcome this fault, we introduce the divided-differences, and give the Newton interpolation.
Divided-difference methods introduced in this section are used to successively generate the polynomials themselves. Our treatment of divided-difference methods will be brief since the results in this section will not be used extensively in subsequent material. Most older texts on numerical analysis have extensive treatment of divided-difference methods. If a more comprehensive treatment is needed, the book by Hildebrand is a particularly good reference.
Suppose that $P_n(x)$ is the $n$th Lagrange polynomial that agrees with the function $f$ at the distinct numbers $x_0, x_1, \cdots, x_n$ are used to express $P_n(x)$ in the form

$$P_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}),$$

for appropriate constants $a_0, a_1, \cdots, a_n$. To determine the first of these constants, $a_0$, note that if $P_n(x)$ is written in the form of Equation above, then evaluating $P_n(x)$ at $x_0$ leaves only the constant term $a_0$; that is,

$$a_0 = P_n(x_0) = f(x_0)$$
Similar, when $P_n(x)$ is evaluated at $x_1$, the only nonzero terms in the evaluation of $P_n(x_1)$ are the constant and linear terms,

$$f(x_0) + a_1(x_1 - x_0) = P_n(x_1) = f(x_1)$$

so

$$a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$
Definition

(\textit{Divided-Difference}) Suppose that the values of the function $f$ at the distinct numbers $x_0, x_1, \cdots, x_n$ are given. Then

$$f \left[ x_i, x_j \right] = \frac{f(x_j) - f(x_i)}{x_j - x_i}, \quad i \neq j,$$

(5.4.1)

is called the \textbf{first divided difference} of $f$ with respect to $x_i$ and $x_j$. 

Fan Yang: yfggd114@163.com
Definition

\[ f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_i, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_{i+1}} \]  \hspace{1cm} (5.4.2)

is called the second divided difference of \( f \) with respect to \( x_i, x_{i+1} \) and \( x_{i+2} \).
Definition

Generally,

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

is called the \textbf{kth divided difference} of \(f\) with respect to \(x_i, x_{i+1}, \cdots, x_{i+k-1}, x_{i+k}\).
Note

The zeroth divided difference of $f$ with respect to $x_i$, for $i = 0, 1, \cdots, n$, denoted $f[x_i]$, is simply the value of $f$ at $x_i$:

$$f[x_i] = f(x_i). \quad (5.4.4)$$
Theorem

(Properties of the Divided-Difference).

(a) If $F(x) = Cf(x)$, where $C$ is a constant, then

$$F [x_0, x_1, \cdots, x_k] = Cf [x_0, x_1, \cdots, x_k].$$  \hspace{1cm} (5.4.5)

(b) If $F(x) = f(x) + g(x)$, then

$$F [x_0, x_1, \cdots, x_k] = f [x_0, x_1, \cdots, x_k] + g [x_0, x_1, \cdots, x_k].$$  \hspace{1cm} (5.4.6)
Theorem

(c) The kth divided difference

\[ f [x_0, x_1, \cdots, x_k] = \sum_{i=0}^{k} \frac{f(x_i)}{\omega_k'(x_i)}, \quad (5.4.8) \]

where \( \omega_k(x) = (x - x_0)(x - x_1) \cdots (x - x_k), \)
\( \omega_k'(x_i) = (x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_k). \)
Theorem

(d) If the sequence $m_0, m_1, \cdots, m_k$ is a permutation of $0, 1, \cdots, k$, then

$$f [x_0, x_1, \cdots, x_k] = f [x_{m_0}, x_{m_1}, \cdots, x_{m_k}], \quad (5.4.9)$$

which means that the value of $f [x_0, x_1, \cdots, x_k]$ is independent of the order of the numbers $x_0, x_1, \cdots, x_n$. 

Fan Yang: yfggd114@163.com
Newton's interpolating polynomial

If \( x_0, x_1, \cdots, x_n \) are \( n + 1 \) distinct numbers and \( y_i = f(x_i) (i = 0, 1, \ldots, n) \), due to the definition of differences formula, as \( x \neq x_i (i = 0, 1, \ldots, n) \), we obtain

\[
f[x_0, x] = \frac{f(x) - f(x_0)}{x - x_0} \Rightarrow f(x) = f(x_0) + f[x_0, x](x - x_0),
\]

\[
f[x_0, x_1, x] = \frac{f[x_0, x] - f[x_0, x_1]}{x - x_1} \Rightarrow f[x_0, x] = f[x_0, x_1] + f[x_0, x_1, x](x - x_1).
\]

Thus

\[
f(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x](x - x_0)(x - x_1).
\]
So
\[ f(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\
+ \ldots + f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1)\ldots(x - x_{n-1}) \\
+ f[x_0, x_1, \ldots, x_n, x](x - x_0)\ldots(x - x_{n-1})(x - x_n) \\
= N_n(x) + R_n(x), \]

here
\[ N_n(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\
+ \ldots + f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1)\ldots(x - x_{n-1}), \]

which is called **Newton’s interpolating polynomial**, 
\[ R_n(x) = f[x_0, x_1, \ldots, x_n, x] \omega_n(x). \]

which is called **Newton’s interpolating remainder term**.

Due to the uniqueness of interpolating polynomial, we know

\[ R_n(x) = f(x) - N_n(x) = f(x) - P_n(x) = \frac{f^{n+1}(\xi)}{(n + 1)!} \omega_n(x). \]

So

\[ f[x_0, x_1, \ldots, x_n, x] = \frac{f^{n+1}(\xi)}{(n + 1)!}. \]
Table 5.4.1 (Divided-Difference Table)

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$f(x_i)$</th>
<th>First Divided Difference</th>
<th>Second Divided Difference</th>
<th>Third Divided Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0$</td>
<td>$f(x_0)$</td>
<td>$f[x_0, x_1]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_1$</td>
<td>$f(x_1)$</td>
<td>$f[x_0, x_1]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>$f(x_2)$</td>
<td>$f[x_0, x_2]$</td>
<td>$f[x_0, x_1, x_2]$</td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td>$f(x_3)$</td>
<td>$f[x_0, x_3]$</td>
<td>$f[x_0, x_1, x_3]$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
where

\[
\begin{align*}
    f[x_0, x_1] &= \frac{f(x_1) - f(x_0)}{x_1 - x_0}, \\
    f[x_0, x_2] &= \frac{f(x_2) - f(x_0)}{x_2 - x_0}, \\
    f[x_0, x_3] &= \frac{f(x_3) - f(x_0)}{x_3 - x_0}, \\
    f[x_0, x_1, x_2] &= \frac{f[x_0, x_2] - f[x_0, x_1]}{x_2 - x_1},
\end{align*}
\]
Interpolating Polynomials
  Lagrange Interpolating
  Newton Interpolating
  Hermite Interpolation
  Piecewise Polynomial Interpolation
  Cubic Spline Interpolation
  Least Squares Method
  Least Squares Approximation
  Orthogonal Polynomials

Divided Differences
  Newton's interpolating polynomial

\[ f[x_0, x_1, x_3] = \frac{f[x_0, x_3] - f[x_0, x_1]}{x_3 - x_1}, \]

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

and so on.
Example

Approximate $f(1.5)$ use following data and Newton’s divided-difference formula.

\[x_0 = 0, \quad x_1 = 1, \quad x_2 = 2, \quad x_3 = 3,\]

\[f(x_0) = -7, \quad f(x_1) = -4, \quad f(x_2) = 5, \quad f(x_3) = 26.\]
Contents

1. Interpolating Polynomials
   - Existence and Uniqueness of Interpolating Polynomials

2. Lagrange Interpolating
   - Lagrange Interpolating Polynomials
   - Lagrange Error Terms

3. Newton Interpolation
   - Divided Differences
   - Newton’s interpolating polynomial

4. Hermite Interpolation

5. Piecewise Polynomial Interpolation
   - Runge’s Phenomenon
   - Piecewise Linear Interpolation
   - Piecewise Cubic Hermite Interpolation

6. Cubic Spline Interpolation

Fan Yang: yfggd114@163.com
Lagrange Interpolation only considers the value of function, but some interpolation not only considers the value of function, but also considers the value of derivative function.
In this section we pay our attention to the cubic Hermite interpolation, a commonly use Hermite interpolation, that is, for given

**Definition**

\[ y_0 = f(x_0), \quad y_0' = f'(x_0), \quad y_1 = f(x_1), \quad y_1' = f'(x_1), \]

find a cubic Hermite polynomial \( H_3(x) \) which satisfies

\[ H_3(x_0) = y_0, \quad H_3'(x_0) = y_0', \quad H_3(x_1) = y_1, \quad H_3'(x_1) = y_1'. \]  \( (5.5.2) \)
To give the solution of the cubic Hermite interpolation, we first assume \( x_0 = 0, \ x_1 = 1 \) for simplicity and the interpolation condition in (5.5.2) become

\[
H_3(0) = y_0, \quad H'_3(0) = y'_0, \quad H_3(1) = y_1, \quad H'_3(1) = y'_1. \quad (5.5.3)
\]
Theorem

**The cubic polynomial**

\[ H_3(x) = y_0 \varphi_0(x) + y_1 \varphi_1(x) + y_0' \psi_0(x) + y_1' \psi_1(x) \quad (5.5.4) \]

has the interpolating properties (5.5.3), where

\[ \varphi_0(x) = (x - 1)^2(2x + 1), \quad \varphi_1(x) = x^2(2x - 3), \]
\[ \psi_0(x) = x(x - 1)^2, \quad \psi_1(x) = x^2(x - 1). \quad (5.5.5) \]

The polynomial \( H_3(x) \), defined in (5.5.4), is called the **cubic Hermite interpolating polynomial**, 

Fan Yang: yfggd114@163.com
and the set of polynomials defined in (5.5.5) is called the **cubic Hermite interpolating basis**.

**Proof**

*Construct cubic polynomials* \( \varphi_0(x) \), \( \varphi_1(x) \), \( \psi_0(x) \), \( \psi_1(x) \) *with interpolating properties:*

\[
\begin{align*}
\varphi_0(0) &= 1, \ &\varphi_0(1) = \varphi_0'(0) = \varphi_0'(1) = 0; \\
\varphi_1(1) &= 1, \ &\varphi_1(0) = \varphi_1'(0) = \varphi_1'(1) = 0; \\
\psi_0'(0) &= 1, \ &\psi_0(0) = \psi_0(1) = \psi_0'(1) = 0;
\end{align*}
\]

Fan Yang: yfggd114@163.com
\[ \psi'_1(1) = 1, \; \psi_1(0) = \psi_1(1) = \psi'_1(0) = 0. \] (5.5.6)

Since \( \varphi_0(1) = \varphi'_0(1) = 0 \), \( x_1 = 1 \) is the zero of multiplicity two of the polynomial \( \varphi_0(x) \), so we can write

\[ \varphi_0(x) = (x - 1)^2(a_0 x + b_0). \] (5.5.7)

Since \( \varphi_0(0) = 1 \) and \( \varphi'_0(0) = 0 \), that is,

\[
\left[ (x - 1)^2(a_0 x + b_0) \right]_{x=0} = (0 - 1)^2(a_0 \times 0 + b_0) = 1 \Rightarrow b_0 = 1;
\]

\[
\left[ (x - 1)^2(a_0 x + b_0) \right]_{x=0}' = \left[ a_0(x - 1)^2 + 2(x - 1)(a_0(x) + b_0) \right]_{x=0}
\]
\[ = 0 \Rightarrow a_0 = 2. \]

Substitute \( a_0 = 2 \) and \( b_0 = 1 \) into (5.5.7) and we have

\[ \varphi_0(x) = (x - 1)^2(2x + 1). \]

Similarly, we can get
\[ \varphi_1(x) = x^2(-2x + 3), \quad \psi_0(x) = x(x - 1)^2, \quad \psi_1(x) = x^2(x - 1). \]

It is easy to verify that

\[ H_3(0) = y_0 \varphi_0(0) + y_1 \varphi_1(0) + y_0' \psi_0(0) + y_1' \psi_1(0) = y_0 \varphi_0(0) = y_0 \times 1 = y_0, \]
\[ H_3(1) = y_0 \varphi_0(1) + y_1 \varphi_1(1) + y_0' \psi_0(1) + y_1' \psi_1(1) = y_1 \varphi_1(1) = y_1 \times 1 = y_1, \]
\[ H'_3(0) = y_0 \varphi'_0(0) + y_1 \varphi'_1(0) + y_0' \psi'_0(0) + y_1' \psi'_1(0) = y'_0 \psi'_0(0) = y'_0 \times 1 = y'_0, \]
\[ H'_3(1) = y_0 \varphi'_0(1) + y_1 \varphi'_1(1) + y_0' \psi'_0(1) + y_1' \psi'_1(1) = y'_1 \psi'_1(1) = y'_1 \times 1 = y'_1. \]
Corollary

If given the points $x_0, x_1$, suppose $h = x_1 - x_0$, $\hat{x} = \frac{x - x_0}{h}$, $\hat{f}(\hat{x}) = f(x)$

$\hat{f}(0) = f(x_0) = y_0$, $\hat{f}(1) = f(x_0) = y_1$, $\hat{f}'(\hat{x}) = f'(x) \frac{dx}{dx} = f'(x)h$

$\hat{f}'(0) = hf'(x_0) = hy_0'$, $\hat{f}'(1) = hf'(x_0) = hy_1'$

Using basic function, we can let

$$H_3(x) = y_0 \varphi_0(\hat{x}) + y_1 \varphi_1(\hat{x}) + y_0' h \psi_0(\hat{x}) + y_1' h \psi_1(\hat{x}). \quad (5.5.8)$$
If $f \in C^4[a, b]$, then

$$f(x) = H_3(x) + \frac{f^{(4)}(\xi)}{4!}(x - x_0)^2(x - x_1)^2,$$  \hspace{1cm} (5.5.10)

for some $\xi$ with $a < \xi < b$, where $H_3(x)$ is defined in (5.5.8).
Example

Construct the Hermite polynomial to agree with the following data

\[ f(1) = 1, \quad f'(1) = 0.5, \quad f(2) = 2.5, \quad f'(2) = 0.8. \]
Solution

\[ H_3(x) = -1.7x^3 + 7.8x^2 - 10x + 4.9. \]
Construct the Hermite polynomial to agree with the following data

\[ f(1) = 2, \quad f'(1) = 1, \quad f(2) = 3, \quad f'(2) = -1. \]
Solution

\[ H_3(x) = -2x^3 + 8x^2 - 9x + 5. \]
Contents

1 Interpolating Polynomials
   - Existence and Uniqueness of Interpolating Polynomials
2 Lagrange Interpolating
   - Lagrange Interpolating Polynomials
   - Lagrange Error Terms
3 Newton Interpolation
   - Divided Differences
   - Newton’s interpolating polynomial
4 Hermite Interpolation
5 Piecewise Polynomial Interpolation
   - Runge’s Phenomenon
   - Piecewise Linear Interpolation
   - Piecewise Cubic Hermite Interpolation
6 Cubic Spline Interpolation

Fan Yang: yfggd114@163.com
Consider the function

\[ f(x) = \frac{1}{1 + 25x^2}. \]

Runge found that if the function is interpolated at equidistant points \( x_i \) between \(-1\) and \(1\) such that

\[ x_i = -1 + i \times \frac{2}{n}, \text{ for } i = 0, 1, \ldots, n, \]

with a polynomial \( p_n(x) \) of degree \( \leq n \), the resulting interpolation oscillates toward the end of the interval, i.e. close to \(-1\) and \(1\).
Runge’s Phenomenon

This phenomenon is called **Runge’s Phenomenon**. In the mathematical field of numerical analysis, Runge’s Phenomenon is a problem that occurs using polynomial interpolation with polynomials of high degree. It was discovered by *Carl David Tolmé Runge* in 1901 when exploring the behavior of errors using polynomial interpolation to approximate certain functions.
An alternative approach 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 \textit{piecewise polynomial approximation}. 
**Definition**

*Give a set of nodes*

\[ a = x_0 < x_1 < \cdots < x_n = b, \]

*and a function f defined on \([a, b]\), whose values are given at these numbers, that is,*

\[ y_i = f(x_i), \text{ for each } i = 0, 1, \cdots, n. \]
A \textit{piecewise linear interpolating polynomial} \( A_1(x) \) for \( f \) is a piecewise polynomial

\[ A_1(x) = y_i \frac{x - x_{i+1}}{x_i - x_{i+1}} + y_{i+1} \frac{x - x_i}{x_{i+1} - x_i}, \quad (5.6.1) \]

where \( x \in [x_i, x_{i+1}] \) for each \( i = 0, 1, \cdots, n - 1 \).
Theorem

Suppose \( f \in C^2[a, b] \). If \( A_1(x) \) given in (5.6.1) is the piecewise linear interpolating polynomial for \( f \) on the nodes

\[
a = x_0 < x_1 < \cdots < x_n = b,
\]

then, for any \( x \in [a, b] \),

\[
|f(x) - A_1(x)| \leq \frac{h^2}{8} M_2, \tag{5.6.2}
\]

where
Interpolating Polynomials
Lagrange Interpolating
Newton Interpolation
Hermite Interpolation
Piecewise Polynomial Interpolation
Cubic Spline Interpolation
Least Squares Method
Least Squares Approximation
Orthogonal Polynomials
Runge's Phenomenon
Piecewise Linear Interpolation
Piecewise Cubic Hermite Interpolation

\[ h = \max_{0 \leq i \leq n-1} |x_{i-1} - x_i| \quad \text{and} \quad M_2 = \max_{a \leq x \leq b} |f''(x)|, \]

and \( A_1(x) \) converges uniformly to \( f(x) \), as \( h \) tends to zero.

**Note**

A disadvantage of piecewise-linear polynomial approximation is that there is likely no differentiability at the endpoints of the subintervals, which, in a geometrical context, means that the interpolating function is not "smooth".

Fan Yang: yfggd114@163.com
Often it is clear from physical conditions that smoothness is required, so the approximating function must be continuously differentiable. An alternative procedure is to use a piecewise polynomial of Hermite type.
A piecewise cubic Hermite interpolating polynomial for \( f \) is a piecewise polynomial 

\[ A_3(x) \] for \( f \) is a piecewise polynomial
\[ A_3(x) = y_i H_{i,i}(x) + y_{i+1} H_{i,i+1}(x) + y_i' \hat{H}_{i,i}(x) + y_{i+1}' \hat{H}_{i,i+1}(x), \]

where \( x \in [x_i, x_{i+1}] \) for each \( i = 0, 1, \cdots, n - 1 \), and

\[
H_{i,i}(x) = \left[ 1 + 2 \left( \frac{x - x_i}{h_i} \right) \right] \left( \frac{x - x_{i+1}}{h_i} \right)^2,
\]

\[
\hat{H}_{i,i} = (x - x_i) \left( \frac{x - x_{i+1}}{h_i} \right)^2.
\]
Interpolating Polynomials
Lagrange Interpolating
Newton Interpolation
Hermite Interpolation
Piecewise Polynomial Interpolation
Cubic Spline Interpolation
Least Squares Method
Least Squares Approximation
Orthogonal Polynomials

Runge’s Phenomenon
Piecewise Linear Interpolation
Piecewise Cubic Hermite Interpolation

\[ H_{i,i+1}(x) = \left[ 1 - 2 \left( \frac{x - x_{i+1}}{h_i} \right) \right] \left( \frac{x - x_i}{h_i} \right)^2, \]

\[ \hat{H}_{i,i+1} = (x - x_{i+1}) \left( \frac{x - x_i}{h_i} \right)^2. \quad (5.6.4) \]

where \( h = x_{i+1} - x_i \) for each \( i = 0, 1, \ldots, n - 1. \)

**Theorem**

*Suppose \( f \in C^4[a, b]. If A_3(x) given in (5.6.3) is a piecewise cubic Hermite interpolating polynomial.*
for $f$ on the nodes

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

then, for any $x \in [a, b]$,

$$|f(x) - A_3(x)| \leq \frac{h^4}{384} M_4,$$  \hspace{1cm} (5.6.5)

where

$$h = \max_{0 \leq i \leq n-1} |x_{i+1} - x_i| \quad \text{and} \quad M_4 = \max_{a \leq x \leq b} \left| f^{(4)}(x) \right|,$$

and $A_3(x)$ converges uniformly to $f(x)$, as $h$ tends to zero.
Contents

1 Interpolating Polynomials
   • Existence and Uniqueness of Interpolating Polynomials

2 Lagrange Interpolating
   • Lagrange Interpolating Polynomials
   • Lagrange Error Terms

3 Newton Interpolation
   • Divided Differences
   • Newton’s interpolating polynomial

4 Hermite Interpolation

5 Piecewise Polynomial Interpolation
   • Runge’s Phenomenon
   • Piecewise Linear Interpolation
   • Piecewise Cubic Hermite Interpolation

6 Cubic Spline Interpolation

Interpolation
Definition of Cubic Spline

**Definition**

Given a function $f$ defined on $[a, b]$ and a division of $[a, b]$:

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

a cubic spline interpolant $S$ for $f$ is a function that satisfies the following condition:
Definition of Cubic Spline
The Construction of Cubic Spline

Definition

(a) $S(x)$ is a cubic polynomial, denoted $S_i(x)$, on the subinterval $[x_i, x_{i+1}]$ for each $i = 0, 1, \cdots, N - 1$;
(b) $S_i(x_i) = S_{i+1}(x_i)$ for each $i = 1, 2, \cdots, N - 1$;
(c) $S'_i(x_i) = S'_{i+1}(x_i)$ for each $i = 1, 2, \cdots, N - 1$;
(d) $S''_i(x_i) = S''_{i+1}(x_i)$ for each $i = 1, 2, \cdots, N - 1$;
(e) $S(x_i) = f(x_i)$ for each $i = 0, 1, \cdots, N$,
where $x_0, x_1, \cdots, x_N$ are called spline knots.
In \([x_i, x_{i+1}]\), the cubic spline interpolant is a cubic polynomial function, we should determine 4 constants. There are \(N\) small intervals, we should compute \(4N\) constants to determine the cubic spline interpolant polynomial function. But above ((a)-(e)), we only have \(4N – 2\) conditions. Using these \(4N – 2\) conditions to determine \(4N\) constants, we still lack two conditions. So we should add two boundary conditions as follows:
Boundary Condition

(1) The First Boundary Conditions:

\[ S'(a) = y_0' \quad \text{and} \quad S'(b) = y_{N+1}'. \quad (6.1) \]

(2) The Second Boundary Conditions:

\[ S''(a) = f_0'' \quad \text{and} \quad S''(b) = f_N''. \quad (6.2) \]
Boundary Condition

The conditions (6.2) are called the natural boundary conditions if

\[ S''(a) = 0 \quad \text{and} \quad S''(b) = 0, \quad (6.3) \]

and a cubic spline with the natural boundary is called the natural spline.

(3) The Third Boundary Conditions:

\[ S'(a) = S'(b) \quad \text{and} \quad S''(a) = S''(b), \quad (6.4) \]

and these conditions occur when \( y = f(x) \) is a periodic function of period \( b - a \). From the periodicity of \( f(x) \) we have \( S(a) = S(b) \).
The Construction of Cubic Spline

Denote

\[ M_i = S''(x_i) \text{ for } i = 0, 1, \ldots, N, \tag{6.5} \]

and \( h_{i+1} = x_{i+1} - x_i \) for \( i = 0, 2, \ldots, N - 1 \).

From part (a) of Definition 5.7.1 we know that \( S''(x) \), denote \( S''_{i+1}(x) \), is a linear polynomial on the subinterval \([x_i, x_{i+1}]\) for each \( i = 0, 1, \ldots, N - 1 \), denote \( f_i = f(x_i) \), that is,

\[
S''_{i+1}(x) = M_i \frac{x - x_{i+1}}{x_i - x_{i+1}} + M_{i+1} \frac{x - x_i}{x_{i+1} - x_i} = M_i \frac{x_{i+1} - x}{h_{i+1}} + M_{i+1} \frac{x - x_i}{h_{i+1}}. \tag{6.6}
\]
Calculating the integral of (6.6) from $x_i$ to $x$ gives

$$S'_{i+1}(x) = \int_{x_i}^{x} S''_{i+1}(x)\,dx = \frac{M_i}{h_{i+1}} \int_{x_i}^{x} (x_{i+1} - x)\,dx + \frac{M_{i+1}}{h_{i+1}} \int_{x_i}^{x} (x - x_i)\,dx$$

$$= -\frac{M_i}{2h_{i+1}} (x_{i+1} - x)^2 + \frac{M_{i+1}}{2h_{i+1}} (x - x_i)^2 + C_1,$$

(6.7)

where $C_1$ is a constant.
The Construction of Cubic Spline

Calculating the integral of (6.7) from $x_i$ to $x$ gives

$$S_{i+1}(x) = \int_{x_i}^{x} S'_{i+1}(x) \, dx = -\frac{M_i}{2h_{i+1}} \int_{x_i}^{x} (x_{i+1} - x)^2 \, dx$$

$$+ \frac{M_{i+1}}{2h_{i+1}} \int_{x_i}^{x} (x - x_i)^2 \, dx + C_1(x - x_i)$$

$$= \frac{M_i}{6h_{i+1}} (x_{i+1} - x)^3 + \frac{M_{i+1}}{6h_{i+1}} (x - x_i)^3 + C_1(x - x_i) + C_2,$$

(6.8)

where $C_2$ is a constant.
The Construction of Cubic Spline

Substituting $S_{i+1}(x_i) = f_i$ and $S_{i+1}(x_{i+1}) = f_{i+1}$ into (6.8) yields

$$C_1 = -\frac{h_{i+1}}{6}(M_{i+1} - M_i) + \frac{1}{h_{i+1}}(f_{i+1} - f_i),$$

(6.9)

$$C_2 = f_i - \frac{h_{i+1}^2}{6}M_i.$$  

(6.10)

Substituting (6.9) and (6.10) into (6.7) and (6.8) respectively gives

$$S_{i+1}'(x) = -\frac{(x_{i+1} - x)^2}{2h_{i+1}}M_i + \frac{(x - x_i)^2}{2h_{i+1}}M_{i+1} + \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{1}{6}h_{i+1}(M_{i+1} - M_i).$$

(6.11)
The Construction of Cubic Spline

\[ S_{i+1}(x) = \frac{(x_{i+1} - x)^3}{6h_{i+1}} M_i + \frac{(x - x_i)^3}{6h_{i+1}} M_{i+1} + \frac{x - x_i}{h_{i+1}} \left( f_{i+1} - \frac{1}{6} h_{i+1}^2 M_{i+1} \right) 
+ \frac{x_{i+1} - x}{h_{i+1}} \left( f_i - \frac{1}{6} h_{i+1}^2 M_i \right), \quad (i = 0, 1, 2, \ldots, N - 1). \]

(6.12)
The Construction of Cubic Spline

Using (6.11) with condition (c): \( S'_i(x_i) = S'_{i+1}(x_i) \) for each \( i = 1, 2, \ldots, N \), we get

\[
S'_i(x_i) = - \frac{(x_i - x_i)^2}{2h_i} M_{i-1} + \frac{(x_i - x_{i-1})^2}{2h_i} M_i \\
+ \frac{f_i - f_{i-1}}{h_i} - \frac{1}{6} h_i (M_i - M_{i-1}),
\]

\[
= \frac{f_i - f_{i-1}}{h_i} + \frac{1}{3} h_i M_i + \frac{1}{6} h_i M_{i-1}.
\] (6.13)
The Construction of Cubic Spline

\[ S'_{i+1}(x) = - \frac{(x_{i+1} - x_i)^2}{2h_{i+1}} M_i + \frac{(x_i - x_i)^2}{2h_{i+1}} M_{i+1} \]

\[ + \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{1}{6} h_{i+1}(M_{i+1} - M_i) \]

\[ = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{1}{3} h_{i+1} M_i - \frac{1}{6} h_{i+1} M_{i+1}. \] (6.14)
The Construction of Cubic Spline

Due to $S'_i(x_i) = S'_{i+1}(x_i)$, using (6.13) and (6.14), we obtain

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = g_i, \quad i = 1, 2, \cdots, N - 1,$$

(6.15)

where

$$\mu_i = \frac{h_i}{h_i + h_{i+1}}, \quad \lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}} = 1 - \mu_i,$$

(6.16)

and

$$g_i = \frac{6}{h_i + h_{i+1}} \left( \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \right).$$

(6.17)
The Construction of Cubic Spline

(1) **Cubic Spline with the First Boundary Conditions**

(6.11) with the first boundary conditions (6.1):

\[ S'(a) = f'_0, \quad S'(b) = f'_N, \]

\[ S''(a) = \lambda_0 M_1, \quad S''(b) = \mu_N M_{N-1}, \]

which gives

\[
2M_0 + \lambda_0 M_1 = g_0, \quad \mu_N M_{N-1} + 2M_N = g_N, \quad (6.18)
\]

where

\[
\lambda_0 = 1, \quad \mu_N = 1,
\]

\[
g_0 = \frac{6}{h_1} \left( \frac{f_1 - f_0}{h_1} - f'_0 \right), \quad g_N = \frac{6}{h_N} \left( f'_N - \frac{f_N - f_{N-1}}{h_N} \right).
\]
Combining (6.15) and (6.18) gives the linear system

\[
\begin{align*}
2M_0 + \lambda_0 M_1 &= g_0, \\
\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} &= g_i, \quad i = 1, 2, \ldots, N - 1, \\
\mu_N M_{N-1} + 2M_N &= g_N,
\end{align*}
\]

(6.19)
The Construction of Cubic Spline

where

\[ \lambda_0 = \mu_N = 1, \quad \mu_i = \frac{h_i}{h_i + h_{i+1}}, \quad \lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}} = 1 - \mu_i, \quad i = 1, 2, \ldots, N - 1. \]

\[ g_0 = \frac{6}{h_1} \left( f_1 - f_0 \right) - f'_0, \quad g_N = \frac{6}{h_N} \left( f'_N - \frac{f_N - f_{N-1}}{h_N} \right), \]

\[ g_i = \frac{6}{h_i + h_{i+1}} \left( \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \right), \quad i = 1, 2, \ldots, N - 1. \]
The linear system (6.19) can be expressed in matrix form

\[
\begin{bmatrix}
2 & \lambda_0 \\
\mu_1 & 2 & \lambda_1 \\
\mu_2 & 2 & \lambda_2 \\
\cdot & \cdot & \cdot \\
\mu_{N-1} & 2 & \lambda_{N-1} \\
\mu_N & 2 &
\end{bmatrix}
\begin{bmatrix}
M_0 \\
M_1 \\
M_2 \\
\vdots \\
M_{N-1} \\
M_N
\end{bmatrix}
=
\begin{bmatrix}
g_0 \\
g_1 \\
g_2 \\
\vdots \\
g_{N-1} \\
g_N
\end{bmatrix}.
\]

(6.20)
(2) **Cubic Spline with the Second Boundary Conditions**

Substituting $S''(a) = M_0 = f''_0$ and $S''(b) = M_N = f''_N$ into (6.15) yields

$$2M_1 + \lambda_1 M_2 = g_1 - \mu_1 f''_0, \quad \mu_{N-1} M_{N-2} + 2M_{N-1} = g_{N-1} - \lambda_{N-1} f''_N.$$  

(6.21)
Combining (6.15) and (6.21) gives the linear system

\[
\begin{cases}
2M_1 + \lambda_1 M_2 = g_1 - \mu_1 f''_0, \\
\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = g_i, & i = 2, 3, \ldots, N - 2, \\
\mu_{N-1} M_{N-2} + 2M_{N-1} = g_{N-1} - \lambda_{N-1} f''_N,
\end{cases}
\]

(6.22)
The Construction of Cubic Spline

where

$$
\mu_i = \frac{h_i}{h_i + h_{i+1}}, \quad \lambda_i = 1 - \mu_i,
$$

$$
g_i = \frac{6}{h_i + h_{i+1}} \left( \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \right), \quad i = 1, 2, \ldots, N - 1.
$$
The matrix form of the linear system (6.22) is

\[
\begin{bmatrix}
2 & \lambda_1 \\
\mu_2 & 2 & \lambda_2 \\
\mu_3 & 2 & \lambda_3 \\
\vdots & \ddots & \ddots \\
\mu_{N-2} & 2 & \lambda_{N-2} \\
\mu_{N-1} & 2 & \lambda_{N-1}
\end{bmatrix}
\begin{bmatrix}
M_1 \\
M_2 \\
M_3 \\
\vdots \\
M_{N-2} \\
M_{N-1}
\end{bmatrix}
= 
\begin{bmatrix}
g_1 - \mu_1 f''_0 \\
g_2 \\
g_3 \\
\vdots \\
g_{N-2} \\
g_{N-1} - \lambda_{N-1} f''_N
\end{bmatrix}
\]

(6.23)
The Construction of Cubic Spline

(3) **Cubic Spline with the Third Boundary Conditions**

\[ s'_0(x_0) = s'_N(x_N), M_0 = s''_0(x_0) = s''_N(x_N) = M_N. \] (6.24)

Taking \( i = 0 \) and \( i = N - 1 \) into (6.11) respectively gives

\[
\begin{align*}
  s'_0(x_0) &= -\frac{(x_1 - x_0)^2}{2h_1}M_0 + \frac{(x_0 - x_0)^2}{2h_1}M_1 + \frac{f_1 - f_0}{h_1} - \frac{1}{6}h_1(M_1 - M_0) \\
  &= -\frac{1}{2}h_1M_0 + \frac{f_1 - f_0}{h_1} - \frac{1}{6}h_1M_1 + \frac{1}{6}h_1M_0 \\
  &= \frac{f_1 - f_0}{h_1} - \frac{1}{6}h_1M_1 - \frac{1}{3}h_1M_0. \quad (6.25)
\end{align*}
\]
The Construction of Cubic Spline

\[
s'_N(x_N) = -\frac{(x_N - x_N)^2}{2h_N} M_{N-1} + \frac{(x_N - x_{N-1})^2}{2h_N} M_N
\]

\[
+ \frac{f_N - f_{N-1}}{h_N} - \frac{1}{6} h_N (M_N - M_{N-1})
\]

\[
= \frac{f_N - f_{N-1}}{h_N} + \frac{1}{6} h_N M_{N-1} + \frac{1}{3} h_N M_N. \tag{6.26}
\]
The Construction of Cubic Spline

Combining (6.24), (6.25) with (6.26), we obtain

\[
\frac{1}{3}h_1 M_0 + \frac{1}{6}h_1 M_1 + \frac{1}{6}h_N M_{N-1} + \frac{1}{3}h_N M_N = \frac{f_1 - f_0}{h_1} - \frac{f_N - f_{N-1}}{h_N}.
\]

(6.27)
The Construction of Cubic Spline

From $M_0 = M_N$, we have

$$\frac{h_1}{h_1 + h_N} M_1 + \frac{h_N}{h_1 + h_N} M_{N-1} + 2M_N = \frac{6}{h_1 + h_N} \left( \frac{f_1 - f_0}{h_1} - \frac{f_N - f_{N-1}}{h_N} \right).$$

(6.28)

Let $\mu_N = \frac{h_N}{h_1 + h_N}$, $\lambda_N = 1 - \mu_N$, $g_N = \frac{f_1 - f_0}{h_1} - \frac{f_N - f_{N-1}}{h_N}$, (6.28) can be rewritten as

$$\lambda_N M_1 + \mu_N M_{N-1} + 2M_N = g_N.$$

(6.29)
The Construction of Cubic Spline

Combining (6.15), $M_0 = M_N$ with (6.29), we obtain the linear system

\[
\begin{align*}
2M_1 + \lambda_1 M_1 + \mu_1 M_N &= g_1, \\
\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} &= g_i, \quad i = 2, 3, \ldots, N - 1, \\
\lambda_N M_1 + \mu_N M_{N-1} + 2M_N &= g_N,
\end{align*}
\]

(6.30)
The Construction of Cubic Spline

where

\[ \mu_i = \frac{h_i}{h_i + h_{i+1}}, \quad \lambda_i = 1 - \mu_i, \]

\[ g_i = \frac{6}{h_i + h_{i+1}} \left( \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \right), \quad i = 1, 2, \ldots, N - 1. \]
The Construction of Cubic Spline

We can also write the equation (6.30) in matrix

\[
\begin{bmatrix}
2 & \lambda_1 & & \\
\mu_2 & 2 & \lambda_2 & \\
\mu_3 & 2 & \lambda_3 & \\
\ddots & \ddots & \ddots & \\
\mu_{N-1} & 2 & \lambda_{N-1} & \\
\lambda_N & \mu_N & 2 & \\
\end{bmatrix}
\begin{bmatrix}
\mu_1 \\
M_1 \\
M_2 \\
M_3 \\
\vdots \\
M_{N-1} \\
M_N \\
\end{bmatrix}
= \begin{bmatrix}
g_1 \\
g_2 \\
g_3 \\
\vdots \\
g_{N-1} \\
g_N \\
\end{bmatrix}
\]

(6.31)
The Construction of Cubic Spline

Solving the linear system (6.19) or (6.22) or (6.30) we get the solution \((M_0, M_1, \cdots, M_N)^T\). Substituting the solution into (6.12) implies the cubic spline

\[
S_{i+1}(x) = \frac{(x_{i+1} - x)^3}{6h_{i+1}} M_i + \frac{(x - x_i)^3}{6h_{i+1}} M_{i+1} \\
+ \frac{x - x_i}{h_{i+1}} (f_{i+1} - \frac{1}{6} h_{i+1}^2 M_{i+1}) \\
+ \frac{x_{i+1} - x}{h_{i+1}} (f_i - \frac{1}{6} h_{i+1}^2 M_i). \tag{6.32}
\]
Theorem

Suppose $f \in C^4[a, b]$, and $S$ is the cubic spline interpolant for $f$ on the nodes $x_0, x_1, \cdots, x_{N+1}$ satisfying one of the following sets of boundary conditions:

(1) the first boundary conditions: $S'(a) = y'_0$ and $S'(b) = y'_{N+1}$;
(2) the second boundary conditions: $S''(a) = y''_0$ and $S''(b) = y''_{N+1}$.

Then
Interpolating Polynomials
Lagrange Interpolating
Newton Interpolation
Hermite Interpolation
Piecewise Polynomial Interpolation
Cubic Spline Interpolation
Least Squares Method
Least Squares Approximation
Orthogonal Polynomials

Definition of Cubic Spline
The Construction of Cubic Spline

\[
\max_{a \leq x \leq b} \left| f^{(k)}(x) - S^{(k)}(x) \right| \leq C_k \max_{a \leq x \leq b} \left| f^{(4)}(x) \right| h^{4-k}, \quad k = 0, 1, 2, \tag{5.7.27}
\]

where

\[
h = \max_{1 \leq i \leq N+1} (x_i, x_{i-1}), \quad C_0 = \frac{5}{584}, \quad C_1 = \frac{1}{24} \quad \text{and} \quad C_2 = \frac{3}{8}, \tag{5.7.28}
\]

and \( S^{(k)}(x) \) converges uniformly to \( f^{(k)}(x) \), as \( h \) tends to zeros, where \( k = 0, 1, 2 \).
Example 1 Given values of $f(x)$ as follows:

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>0</th>
<th>1</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x_i)$</td>
<td>0</td>
<td>-2</td>
<td>-8</td>
<td>-4</td>
</tr>
</tbody>
</table>

(1) Construct the cubic spline $S$ with boundary conditions $s'(0) = \frac{5}{2}$ and $s'(5) = \frac{19}{4}$ for the given data.

(2) Evaluate $s(0.5)$, $s(3)$, $s(4.5)$.
Example 2 Given values of \( f(x) \) as follows:

<table>
<thead>
<tr>
<th>( x_i )</th>
<th>-1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(x_i) )</td>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( f'(x_i) )</td>
<td>0</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

Construct the cubic spline \( S \).
Example 3 Given values of $f(x)$ as follows:

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x_i)$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$f'(x_i)$</td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$f''(x_i)$</td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Construct the cubic spline $S$ under the first and second boundary conditions, respectively.
Contents

1. Interpolating Polynomials
   - Existence and Uniqueness of Interpolating Polynomials
2. Lagrange Interpolating
   - Lagrange Interpolating Polynomials
   - Lagrange Error Terms
3. Newton Interpolation
   - Divided Differences
   - Newton’s interpolating polynomial
4. Hermite Interpolation
5. Piecewise Polynomial Interpolation
   - Runge’s Phenomenon
   - Piecewise Linear Interpolation
   - Piecewise Cubic Hermite Interpolation
6. Cubic Spline Interpolation

Fan Yang: yfggd114@163.com
Definition

The set of functions \( \varphi_0(x), \varphi_1(x), \cdots, \varphi_n(x) \) is said to be **linearly independent** on \([a, b]\) if, whenever

\[
k_0 \varphi_0(x) + k_1 \varphi_1(x) + \cdots + k_n \varphi_n(x) = 0, \quad (5.8.1)
\]

for all \( x \in [a, b] \), we have \( k_0 = k_1 = \cdots = k_n = 0 \). Otherwise the set of functions is said to be **linearly dependent**.
The set of functions, denoted

\[ \Phi = \text{span} \{ \varphi_0, \varphi_1, \cdots, \varphi_n \}, \quad (5.8.2) \]

is called the **linear space spanned by** \( \varphi_0, \varphi_1, \cdots, \varphi_n \), if every \( \varphi \in \Phi \) can be expressed uniquely as a linear combination of the \( \varphi_i \), that is, for any \( \varphi_0 \in \Phi \), we can find \( c_0, c_1, \cdots, c_n \in \mathbb{R} \), such that

\[ \varphi(x) = c_0 \varphi_0(x) + c_1 \varphi_1(x) + \cdots + c_n \varphi_n(x). \quad (5.8.3) \]

If \( \varphi_0, \varphi_1, \cdots, \varphi_n \) are linearly independent on \([a, b]\), then \( \{ \varphi_0, \varphi_1, \cdots, \varphi_n \} \) is called a **basis** of \( \Phi \).
**Problem**  Given a set of data \((x_i, y_i)\), for \(i = 0, 1, \cdots, m(m \gg n)\). The problem is to find a function \(\varphi^* \in \Phi\), such that

\[
\sum_{i=0}^{m} \rho_i [y_i - \varphi^*(x_i)]^2 = \min_{\varphi \in \Phi} \sum_{i=0}^{m} \rho_i [y_i - \varphi(x_i)]^2,
\]

where \(\rho_i\) is the weight for each \(i = 0, 1, \cdots, m\).
The technique for finding the function $\varphi^*$ is called the \textit{discrete least squares method}; $\varphi^*$ is the \textit{least squares solution} to the problem, and is called the \textit{least squares approximation function with respect to the weight} $\rho_i$; and the linear space $\Phi$, defined in (5.8.2), is called the \textit{class of fitting functions}. In this section, we mainly introduce the case that $\varphi_i(x) = x^i$ for $i = 0, 1, \cdots, n$, which are linearly independent, and therefore
\[ \Phi = \text{span} \{ \varphi_0, \varphi_1, \cdots, \varphi_n \} = \mathbb{P}_n, \]

the collection of polynomials of degree \( \leq n \). So the problem becomes to find a polynomial

\[ p_n(x) = \sum_{k=0}^{n} c_k x^k \in \mathbb{P}_n, \quad (5.8.5) \]
called the **least squares approximation function with respect to the weight** $\rho_i$, to minimize the total error

$$Q = \sum_{i=0}^{m} \rho_i [y_i - p_n(x_i)]^2 = \sum_{i=0}^{m} \rho_i [y_i - \sum_{k=0}^{n} c_k x_i^k]^2. \quad (5.8.6)$$
Solution

For simplicity, we choose $\rho_i = 1$ for $i = 0, 1, \cdots, m$. For minimizing the total error

$$ Q = Q(c_0, c_1, \cdots, c_n) $$

$$ = \sum_{i=0}^{m} \left[ y_i - (c_0 + c_1 x_i + c_2 x_i^2 + \cdots + c_n x_i^n) \right]^2, $$

it is necessary that

Fan Yang: yfggd114@163.com
\[ \frac{\partial Q}{\partial c_i} = 0, \text{ for each } i = 0, 1, \cdots, n, \]

that is,

\[
\begin{cases}
  \frac{\partial Q}{\partial c_0} = 2 \sum_{i=0}^{m} [y_i - (c_0 + c_1 x_i + c_2 x_i^2 + \cdots + c_n x_i^n)](-1) = 0, \\
  \frac{\partial Q}{\partial c_1} = 2 \sum_{i=0}^{m} [y_i - (c_0 + c_1 x_i + c_2 x_i^2 + \cdots + c_n x_i^n)](-x_i) = 0,
\end{cases}
\]
\[
\begin{align*}
\frac{\partial Q}{\partial c_2} &= 2 \sum_{i=0}^{m} [y_i - (c_0 + c_1 x_i + c_2 x_i^2 + \cdots + c_n x_i^n)](-x_i^2) = 0, \\
&\quad \vdots \\
\frac{\partial Q}{\partial c_n} &= 2 \sum_{i=0}^{m} [y_i - (c_0 + c_1 x_i + c_2 x_i^2 + \cdots + c_n x_i^n)](-x_i^n) = 0.
\end{align*}
\]
These equations are simplified to the system of equations

\[
\begin{cases}
    c_0 \times (m + 1) + c_1 \sum_{i=0}^{m} x_i + c_2 \sum_{i=0}^{m} x_i^2 + \cdots + c_n \sum_{i=0}^{m} x_i^n \\
    = \sum_{i=0}^{m} y_i,

    c_0 \sum_{i=0}^{m} x_i + c_1 \sum_{i=0}^{m} x_i^2 + c_2 \sum_{i=0}^{m} x_i^3 + \cdots + c_n \sum_{i=0}^{m} x_i^{n+1} \\
    = \sum_{i=0}^{m} x_i y_i,
\end{cases}
\]
Interpolating Polynomials
Lagrange Interpolating
Newton Interpolation
Hermite Interpolation
Piecewise Polynomial Interpolation
Cubic Spline Interpolation
Least Squares Method
Least Squares Method in Inner Product
Orthogonal Polynomials

\[
\begin{align*}
    c_0 \sum_{i=0}^{m} x_i^2 + c_1 \sum_{i=0}^{m} x_i^3 + c_2 \sum_{i=0}^{m} x_i^4 + \cdots + c_n \sum_{i=0}^{m} x_i^{n+2} \\
    = \sum_{i=0}^{m} x_i^2 y_i,
\end{align*}
\]

\[
\begin{align*}
    &\vdots
\end{align*}
\]

\[
\begin{align*}
    c_0 \sum_{i=0}^{m} x_i^n + c_1 \sum_{i=0}^{m} x_i^{n+1} + c_2 \sum_{i=0}^{m} x_i^{n+2} + \cdots + c_n \sum_{i=0}^{m} x_i^{2n} \\
    = \sum_{i=0}^{m} x_i^n y_i,
\end{align*}
\]

which is a linear system in the \( n+1 \) unknowns \( c_0, c_1, \cdots, c_n \).
The linear system defined in (5.8.7) can be expressed in matrix form

\[
\begin{bmatrix}
    m + 1 & \cdots & \sum_{i=0}^{m} x_i^n \\
    \sum_{i=0}^{m} x_i & \cdots & \sum_{i=0}^{m} x_i^{n+1} \\
    \sum_{i=0}^{m} x_i^2 & \cdots & \sum_{i=0}^{m} x_i^{n+2} \\
    \vdots & \ddots & \vdots \\
    \sum_{i=0}^{m} x_i^n & \cdots & \sum_{i=0}^{m} x_i^{2n}
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
c_2 \\
\vdots \\
c_n
\end{bmatrix}
= \begin{bmatrix}
    \sum_{i=0}^{m} y_i \\
    \sum_{i=0}^{m} x_i y_i \\
    \sum_{i=0}^{m} x_i^2 y_i \\
    \vdots \\
    \sum_{i=0}^{m} x_i^n y_i
\end{bmatrix}.
\]
Example

Find a least squares polynomial of degree 2 with respect to the weights $\rho_i=1$ to fit the data in Table 5.8.1, and compute the total error.

Table 5.8.1

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i$</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
<td>0.75</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>$y_i$</td>
<td>1.152</td>
<td>1.543</td>
<td>1.975</td>
<td>2.221</td>
<td>2.279</td>
<td>2.311</td>
</tr>
</tbody>
</table>
Solution

For this problem, $n = 2$, $m = 5$. (5.8.8) gives

$$
\begin{bmatrix}
6.0000 & 3.5500 & 2.7225 \\
3.5500 & 2.7225 & 2.3039 \\
2.7225 & 2.3039 & 2.0432
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
c_2
\end{bmatrix}
= 
\begin{bmatrix}
11.481 \\
7.5935 \\
6.0504
\end{bmatrix}.
$$
We solve this system and obtain

\[ c_0 = 0.8593, \quad c_1 = 2.8745, \quad \text{and} \quad c_2 = -1.4252. \]

Thus, the least squares polynomial of degree 2 fitting the proceeding data is

\[ p_2(x) = c_0 + c_1 x + c_2 = 0.8593 + 2.8745x - 1.4252x^2. \]

At the given values of \( x_i \) we have the approximations shown in Table 5.8.2.
### Table 5.8.2

| $i$ | $x_i$ | $y_i$ | $p_2(x_i)$ | $|y_i - p_2(x_i)|$ |
|-----|-------|-------|------------|------------------|
| 0   | 0.1   | 1.152 | 1.133      | $1.944 \times 10^{-2}$ |
| 1   | 0.3   | 1.543 | 1.593      | $5.048 \times 10^{-2}$ |
| 2   | 0.5   | 1.975 | 1.940      | $3.462 \times 10^{-2}$ |
| 3   | 0.75  | 2.221 | 2.214      | $7.334 \times 10^{-2}$ |
| 4   | 0.9   | 2.279 | 2.292      | $1.313 \times 10^{-2}$ |
| 5   | 1.0   | 2.311 | 2.309      | $2.198 \times 10^{-2}$ |
The total error

\[ Q = Q(c_0, c_1, c_2) = \sum_{i=0}^{5} [y_i - p_2(x_i)]^2 = 4.356 \times 10^{-3}. \]
Definition

A complex vector space $H$ is called an **inner product space** if to each ordered pair of vectors $x$ and $y \in H$ there is associated a complex number $(x, y)$, the so-called **inner product** (or **scalar product**) of $x$ and $y$, such that the following rules hold:

$$(a)(y, x) = (x, y).$$  (The bar denotes complex conjugation.)
(b) \((x + y, z) = (x, z) + (y, z)\) if \(x, y\) and \(z \in H\).
(c) \((\alpha x, y) = \alpha (x, y)\) if \(x\) and \(y \in H\) and \(\alpha\) is a scalar.
(d) \((x, x) \geq 0\) for all \(x \in H\).
(e) \((x, x) = 0\) only if \(x = 0\).

Two vectors \(x\) and \(y\) are said to be **orthogonal**, if \((x, y) = 0\).
Theorem

For any two functions $f$ and $g$, the operation

$$(f, g) = \sum_{i=1}^{m} \rho_i f(x_i) g(x_i) \quad (5.8.10)$$

is the inner product of $f$ and $g$ with respect to the weight $\rho_i$ on $x_1, x_2, \cdots, x_m$, where $\rho_i \geq 0$, for $i = 1, 2, \cdots, m$. With the notation in (5.8.10), the linear system in (5.8.8) can be written in inner product form.
Interpolating Polynomials
Lagrange Interpolating
Newton Interpolation
Hermite Interpolation
Piecewise Polynomial Interpolation
Cubic Spline Interpolation
Least Squares Method
Least Squares Approximation
Orthogonal Polynomials

\[
\begin{bmatrix}
(\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & (\varphi_0, \varphi_2) & \cdots & (\varphi_0, \varphi_n) \\
(\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & (\varphi_1, \varphi_2) & \cdots & (\varphi_1, \varphi_n) \\
(\varphi_2, \varphi_0) & (\varphi_2, \varphi_1) & (\varphi_2, \varphi_2) & \cdots & (\varphi_2, \varphi_n) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
(\varphi_n, \varphi_0) & (\varphi_n, \varphi_1) & (\varphi_n, \varphi_2) & \cdots & (\varphi_n, \varphi_n)
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
c_2 \\
\vdots \\
c_n
\end{bmatrix}
= \begin{bmatrix}
(\varphi_0, f) \\
(\varphi_1, f) \\
(\varphi_2, f) \\
\vdots \\
(\varphi_n, f)
\end{bmatrix},
\]

(5.8.11)
where \( \varphi_j(x) = x^j \) for \( j = 1, 2, \cdots, n \); \( \rho_i = 1, f(x_i) = y, \) for \( i = 1, 2, \cdots, m \);

\[
(\varphi_j, \varphi_k) = \sum_{i=0}^{m} \varphi_j(x_i)\varphi_k(x_i) = \sum_{i=0}^{m} x_i^j x_i^k, \text{ for } j, k = 0, 1, \cdots, n; \tag{5.8.12}
\]

\[
(\varphi_j, f) = \sum_{i=0}^{m} \varphi_j(x_i)f(x_i) = \sum_{i=0}^{m} x_i^j y_i, \text{ for } j = 0, 1, \cdots, n. \tag{5.8.13}
\]
Example

Use the method in (5.8.11) to find a least squares polynomial of degree 2 fitting the data in Table 5.8.3, and compute the total error $Q$.

Table 5.8.3

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i$</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
<td>0.75</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>$y_i$</td>
<td>1.152</td>
<td>1.543</td>
<td>1.975</td>
<td>2.221</td>
<td>2.279</td>
<td>2.311</td>
</tr>
</tbody>
</table>
Solution

For this problem, \( n=2, m=5 \).

\[\varphi_0(x) = 1, \quad \varphi_1(x) = x, \quad \varphi_2(x) = x^2, \quad f(x_i) = y_i,\]

for \( i=0, 1, 2, 3, 4, 5 \).

(5.8.11) gives
Interpolating Polynomials
Lagrange Interpolating
Newton Interpolation
Hermite Interpolation
Piecewise Polynomial Interpolation
Cubic Spline Interpolation
Least Squares Method
Least Squares Approximation
Orthogonal Polynomials

Least Squares Method
Least Squares Method in Inner Product

$\begin{bmatrix}
(\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & (\varphi_0, \varphi_2) \\
(\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & (\varphi_1, \varphi_2) \\
(\varphi_2, \varphi_0) & (\varphi_2, \varphi_1) & (\varphi_2, \varphi_2)
\end{bmatrix}\begin{bmatrix}
C_0 \\
C_1 \\
C_2
\end{bmatrix} = \begin{bmatrix}
(\varphi_0, f) \\
(\varphi_1, f) \\
(\varphi_2, f)
\end{bmatrix},$
that is,

\[
\begin{bmatrix}
  6.0000 & 3.5500 & 2.7225 \\
  3.5500 & 2.7225 & 2.3039 \\
  2.7225 & 2.3039 & 2.0432
\end{bmatrix}
\begin{bmatrix}
  c_0 \\
  c_1 \\
  c_2
\end{bmatrix}
= \begin{bmatrix}
  11.481 \\
  7.5935 \\
  6.0504
\end{bmatrix}.
\]
Solving this system yields

\[ c_0 = 0.8593, \quad c_1 = 2.8745, \quad \text{and} \quad c_2 = -1.4252. \]

Thus, the least squares polynomial of degree 2 fitting the proceeding data is

\[ p_2(x) = c_0 + c_1 x + c_2 x^2 = 0.8593 + 2.8745x - 1.4252x^2, \]

and the total error

\[ Q = Q(c_0, c_1, c_2) = \sum_{i=0}^{5} [y_i - p_2(x_i)]^2 = 4.356 \times 10^{-3}. \]
Example

Find a least squares function fitting the following data, and compute the total error $Q$.

Table 5.8.4

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i$</td>
<td>0.0</td>
<td>0.5</td>
<td>1.0</td>
<td>1.5</td>
<td>2.0</td>
<td>2.5</td>
</tr>
<tr>
<td>$y_i$</td>
<td>2.0</td>
<td>1.2</td>
<td>0.9</td>
<td>0.6</td>
<td>0.4</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Solution

Since the graph of the values in Table 5.8.4 appears that the actual relationship between $x$ and $y$ is exponential, we choose an approximating function of the form

$$
\psi(x) = ae^{bx}, \quad (5.8.14)
$$

to fit the above data, where $a$ and $b$ are constants. The method is to consider the logarithm of approximating function in (5.8.14):

$$
ln\psi(x) = lna + bx. \quad (5.8.15)
$$
Denote

\[ p_1(x) = \ln \psi(x), \quad c_0 = \ln a, \quad c_1 = b \quad \text{and} \quad x = x, \]

then (5.8.15) becomes

\[ p_1(x) = c_0 + c_1 x. \]

For this case, \( n=1, \ m=5, \)
\( \varphi_0(x) = 1, \ \varphi_1(x) = x, \ f(x_i) = \ln y_i, \ for \ i = 0, 1, 2, 3, 4, 5. \) Solving the following linear system

\[
\begin{bmatrix}
(\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) \\
(\varphi_1, \varphi_0) & (\varphi_1, \varphi_1)
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1
\end{bmatrix}
= 
\begin{bmatrix}
(\varphi_0, f) \\
(\varphi_1, f)
\end{bmatrix},
\]

Fan Yang: yfggd114@163.com
that is,
\[
\begin{bmatrix}
6 & 7.5 \\
7.5 & 13.75
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1
\end{bmatrix} =
\begin{bmatrix}
-1.8610 \\
-5.6229
\end{bmatrix},
\]
we obtain \( c_0 = 0.6318 \) \textit{and} \( c_1 = -0.7535 \). Thus,
\[
a = e^{c_0} = 1.8809 \textit{ and } b = c_1 = -0.7535,
\]
and the least squares function fitting the proceeding data is
\[
\psi(x) = ae^{bx} = 11.8809e^{-0.7534x}.
\]
At the given values of $x_i$ we have the approximations shown in Table 5.8.5.

| $i$ | $x_i$ | $y_i$ | $p_2(x_i)$ | $|y_i - p_2(x_i)|$ |
|-----|-------|-------|------------|-------------------|
| 0   | 0.0   | 2.0   | 1.8809     | $1.191 \times 10^{-2}$ |
| 1   | 0.5   | 1.2   | 1.2904     | $9.045 \times 10^{-2}$ |
| 2   | 1.0   | 0.9   | 0.8853     | $1.466 \times 10^{-2}$ |
| 3   | 1.5   | 0.6   | 0.6074     | $7.412 \times 10^{-2}$ |
| 4   | 2.0   | 0.4   | 0.4167     | $1.673 \times 10^{-2}$ |
| 5   | 2.5   | 0.3   | 0.2859     | $1.409 \times 10^{-2}$ |
The total error

$$Q = \sum_{i=0}^{5} [y_i - \psi(x_i)]^2 = 2.311 \times 10^{-2}.$$ 

In general, for the class of fitting functions

$$\Phi = \text{span} \{ \varphi_0, \varphi_1, \cdots, \varphi_n \} ,$$

where $\varphi_0, \varphi_1, \cdots, \varphi_n$ are linearly independent, from the following more general normal system
Interpolating Polynomials
Lagrange Interpolating
Newton Interpolation
Hermite Interpolation
Piecewise Polynomial Interpolation
Cubic Spline Interpolation
Least Squares Method
Least Squares Approximation
Orthogonal Polynomials

\[
\begin{pmatrix}
(\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_n) \\
(\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \cdots & (\varphi_1, \varphi_n) \\
\vdots & \vdots & \ddots & \vdots \\
(\varphi_n, \varphi_0) & (\varphi_n, \varphi_1) & \cdots & (\varphi_n, \varphi_n)
\end{pmatrix}
\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} (\varphi_0, f) \\ (\varphi_1, f) \\ \vdots \\ (\varphi_n, f) \end{pmatrix},
\]

(5.8.16)
where $f(x_i) = y_i$ for $i = 1, 2, \cdots, m$;

\[
(\varphi_j, \varphi_k) = \sum_{i=0}^{m} \rho_i \varphi_j(x_i) \varphi_k(x_i), \text{ for } j, k = 0, 1, \cdots, n; \tag{5.8.17}
\]

and

\[
(\varphi_j, f) = \sum_{i=0}^{m} \rho_i \varphi_j(x_i) f(x_i), \text{ for } j = 0, 1, \cdots, n, \tag{5.8.18}
\]

we can get the least squares function

\[
\varphi^*(x) = c_0 \varphi_0(x) + c_1 \varphi_1(x) + \cdots + c_n \varphi_n(x) \in \Phi, \tag{5.8.19}
\]

fitting the given data $(x_i, f(x_i))$ for $i = 1, 2, \cdots, m$. 

Fan Yang: yfggd114@163.com
Example

Find a least squares function fitting the following data, and compute the total error $Q$.

Table 5.8.6

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i$</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>$y_i$</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Solution

Since the graph of the values in Table 5.8.6 appears that the actual relationship between \(x\) and \(y\) is hyperbolic, we choose the class of fitting functions

\[
\Phi = \text{span} \{ \varphi_0(x), \varphi_1(x) \} = \text{span} \{1, 1/x\},
\]

and find a function \(\varphi^* \in \Phi\):

\[
\varphi^*(x) = c_0\varphi_0(x) + c_1\varphi_1(x) = c_0 \times 1 + c_1 \times \frac{1}{x},
\]

to fit the above data.
Solving the following linear system

\[
\begin{bmatrix}
(\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) \\
(\varphi_1, \varphi_0) & (\varphi_1, \varphi_1)
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1
\end{bmatrix} =
\begin{bmatrix}
(\varphi_0, f) \\
(\varphi_1, f)
\end{bmatrix},
\]

that is,

\[
\begin{bmatrix}
4 & 1.8429 \\
1.8429 & 1.3104
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1
\end{bmatrix} =
\begin{bmatrix}
16 \\
11.5429
\end{bmatrix},
\]
we obtain $c_0 = -0.6154$ and $c_1 = 9.0412$. Thus, and the least squares function fitting the proceeding data is

$$\varphi^*(x) = c_0 \times 1 + c_1 \times \frac{1}{x} = -0.6154 + \frac{9.0412}{x}.$$ 

At the given values of $x_i$ we have the approximations shown in Table 5.8.7.
**Table 5.8.7**

| $i$ | $x_i$ | $y_i$ | $p_2(x_i)$ | $|y_i - p_2(x_i)|$ |
|-----|------|------|------------|------------------|
| 0   | 1    | 9    | 8.8758     | 0.1242           |
| 1   | 2    | 4    | 4.3552     | 0.3552           |
| 2   | 5    | 2    | 1.6428     | 0.3572           |
| 3   | 7    | 1    | 1.1262     | 0.1262           |

The total error

$$Q = \sum_{i=0}^{3} [y_i - \psi^*(x_i)]^2 = 0.2851$$
Contents

1 Interpolating Polynomials
   • Existence and Uniqueness of Interpolating Polynomials

2 Lagrange Interpolating
   • Lagrange Interpolating Polynomials
   • Lagrange Error Terms

3 Newton Interpolation
   • Divided Differences
   • Newton’s interpolating polynomial

4 Hermite Interpolation

5 Piecewise Polynomial Interpolation
   • Runge’s Phenomenon
   • Piecewise Linear Interpolation
   • Piecewise Cubic Hermite Interpolation

6 Cubic Spline Interpolation
For any two integrable functions $f$ and $g$, the operation

$$(f, g) = \int_{a}^{b} \rho(x)f(x)g(x)dx$$

is the inner product of $f$ and $g$ with respect to the weight function $\rho(x)$ on $[a, b]$, where $\rho(x) \geq 0$ for $x \in [a, b]$. 

Theorem
**Problem** Suppose $f \in C[a, b]$. The problem is to find a polynomial

$$p_n(x) = c_0\varphi_0(x) + c_1\varphi_1(x) + \cdots + c_n\varphi_n(x) \in \mathbb{P}_n,$$

(5.9.2)

called the *least squares approximation polynomial with respect to the weight function* $\rho(x)$, to minimize the total error

$$Q = Q(c_0, c_1, \cdots, c_n) = \int_a^b \rho(x)[f(x) - p_n(x)]^2 dx$$
Interpolating Polynomials
Lagrange Interpolating
Newton Interpolation
Hermite Interpolation
Piecewise Polynomial Interpolation
Cubic Spline Interpolation
Least Squares Method
Least Squares Approximation
Orthogonal Polynomials

\[
= \int_{a}^{b} \rho(x) \{ f(x) - [c_0 \varphi_0(x) + c_1 \varphi_1(x) + \cdots + c_n \varphi_n(x)] \}^2 dx
\]

where \( \rho(x) \) is a weight function on \([a, b]\).

**Solution**

*In a similar manner used in section 5.7, for minimizing the total error \( Q = Q(c_0, c_1, \cdots, c_n) \) defined in (5.9.3), it is necessary that*

\[
\frac{\partial Q}{\partial c_i} = 0, \text{ for each } i = 0, 1, \cdots, n,
\]
that is,

\[
\begin{align*}
\frac{\partial Q}{\partial c_0} &= -2 \int_a^b \rho(x) \{ f(x) \\ &- [c_0 \varphi_0(x) + c_1 \varphi_1(x) + \cdots + c_n \varphi_n(x)] \} \varphi_0(x) dx = 0, \\
\frac{\partial Q}{\partial c_1} &= -2 \int_a^b \rho(x) \{ f(x) \\ &- [c_0 \varphi_0(x) + c_1 \varphi_1(x) + \cdots + c_n \varphi_n(x)] \} \varphi_1(x) dx = 0,
\end{align*}
\]
\[
\frac{\partial Q}{\partial c_n} = -2 \int_a^b \rho(x) \{ f(x) - [c_0 \varphi_0(x) + c_1 \varphi_1(x) + \cdots + c_n \varphi_n(x)] \} \varphi_n(x) dx = 0.
\]
These equations are simplified to the system of equations

\[
\begin{aligned}
&c_0 \int_a^b \rho(x)\varphi_0(x)\varphi_0(x)dx + c_1 \int_a^b \rho(x)\varphi_0(x)\varphi_1(x)dx + \cdots \\
&+ c_n \int_a^b \rho(x)\varphi_0(x)\varphi_n(x)dx = \int_a^b \rho(x)\varphi_0(x)f(x)dx, \\
&c_0 \int_a^b \rho(x)\varphi_1(x)\varphi_0(x)dx + c_1 \int_a^b \rho(x)\varphi_1(x)\varphi_1(x)dx + \cdots \\
&+ c_n \int_a^b \rho(x)\varphi_1(x)\varphi_n(x)dx = \int_a^b \rho(x)\varphi_1(x)f(x)dx,
\end{aligned}
\]
\[
\begin{align*}
\quad & c_0 \int_a^b \rho(x) \varphi_n(x) \varphi_0(x) \, dx + c_1 \int_a^b \rho(x) \varphi_n(x) \varphi_1(x) \, dx + \cdots \\
& + c_n \int_a^b \rho(x) \varphi_n(x) \varphi_n \, dx = \int_a^b \rho(x) \varphi_n(x) f(x) \, dx,
\end{align*}
\]

which is a linear system in the \( n+1 \) unknowns \( c_0, c_1, \cdots, c_n \).
The linear system defined in (5.9.4) can be expressed in matrix form as follow

\[
\begin{bmatrix}
\int_a^b \rho(x) \varphi_0(x) \varphi_0(x) \, dx & \cdots & \int_a^b \rho(x) \varphi_0(x) \varphi_n(x) \, dx \\
\int_a^b \rho(x) \varphi_1(x) \varphi_0(x) \, dx & \cdots & \int_a^b \rho(x) \varphi_1(x) \varphi_n(x) \, dx \\
\vdots & \ddots & \vdots \\
\int_a^b \rho(x) \varphi_n \varphi_0 \, dx & \cdots & \int_a^b \rho(x) \varphi_n \varphi_n \, dx
\end{bmatrix}
\]
\[
\begin{bmatrix}
  c_0 \\
  c_1 \\
  \vdots \\
  c_n
\end{bmatrix}
\ast
\begin{bmatrix}
  \int_a^b \rho(x) \varphi_0(x) f(x) \, dx \\
  \int_a^b \rho(x) \varphi_1(x) f(x) \, dx \\
  \vdots \\
  \int_a^b \rho(x) \varphi_n(x) f(x) \, dx
\end{bmatrix}
= 
\begin{bmatrix}
  \int_a^b \rho(x) \varphi_0(x) f(x) \, dx \\
  \int_a^b \rho(x) \varphi_1(x) f(x) \, dx \\
  \vdots \\
  \int_a^b \rho(x) \varphi_n(x) f(x) \, dx
\end{bmatrix}.
\]
The linear system in (5.9.4) or (5.9.5) is called the **normal system of equations**. With the notation in (5.9.1), the linear system in (5.9.5) can be written in inner product form

\[
\begin{bmatrix}
(\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_n) \\
(\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \cdots & (\varphi_1, \varphi_n) \\
\vdots & \vdots & \ddots & \vdots \\
(\varphi_n, \varphi_0) & (\varphi_n, \varphi_1) & \cdots & (\varphi_n, \varphi_n)
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
\vdots \\
c_n
\end{bmatrix}
=
\begin{bmatrix}
(\varphi_0, f) \\
(\varphi_1, f) \\
\vdots \\
(\varphi_n, f)
\end{bmatrix},
\]

(5.9.6)
where

\[(\varphi_j, \varphi_k) = \int_a^b \rho(x) \varphi_j(x) \varphi_k(x) \, dx, \text{ for } j, k = 0, 1, \ldots, n; \quad (5.9.7)\]

and

\[(\varphi_j, f) = \int_a^b \rho(x) \varphi_j(x) f(x) \, dx, \text{ for } j = 0, 1, \ldots, n, \quad (5.9.8)\]

In this section, we only concern the case that \(\varphi_j(x)\) is a polynomial of degree \(j\), for each \(j = 0, 1, \cdots, n\).
**Example**

Find the least squares approximating polynomial of degree 2 for the function \( f(x) = e^x \) on the interval \([0, 1] \). Choose the weight function \( \rho(x) \equiv 1 \).

**Solution**

For this problem, \( n = 2 \),

\[
\varphi_0(x) = 1, \quad \varphi_1(x) = x, \quad \varphi_2(x) = x^2, \quad f(x) = e^x.
\]
With \( \rho(x) \equiv 1 \), (5.9.7) and (5.9.8) give

\[
(\varphi_0, \varphi_0) = \int_0^1 1 \times 1 \, dx = 1, \\
(\varphi_0, \varphi_1) = \int_0^1 1 \times x \, dx = \frac{1}{2}, \\
(\varphi_0, \varphi_2) = \int_0^1 1 \times x^2 \, dx = \frac{1}{3}, \\
(\varphi_1, \varphi_0) = \int_0^1 x \times 1 \, dx = \frac{1}{2}, \\
(\varphi_1, \varphi_1) = \int_0^1 x \times x \, dx = \frac{1}{3}, \\
(\varphi_1, \varphi_2) = \int_0^1 1 \times x^2 \, dx = \frac{1}{4},
\]

Fan Yang: yfggd114@163.com


\[(\varphi_2, \varphi_0) = \int_0^1 x^2 \times 1 \, dx = \frac{1}{3}, (\varphi_2, \varphi_1) = \int_0^1 x^2 \times x \, dx = \frac{1}{4}, \]

\[(\varphi_2, \varphi_2) = \int_0^1 x^2 \times x^2 \, dx = \frac{1}{5}, \]

\[(\varphi_0, f) = \int_0^1 1 \times e^x \, dx = 1.7183, (\varphi_1, f) = \int_0^1 x \times e^x \, dx = 1.0000, \]

\[(\varphi_2, f) = \int_0^1 x^2 \times e^x \, dx = 0.7183. \]
Substituting these values into the following linear system

\[
\begin{bmatrix}
(\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & (\varphi_0, \varphi_2) \\
(\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & (\varphi_1, \varphi_2) \\
(\varphi_2, \varphi_0) & (\varphi_2, \varphi_1) & (\varphi_2, \varphi_2)
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
c_2
\end{bmatrix}
=
\begin{bmatrix}
(\varphi_0, f) \\
(\varphi_1, f) \\
(\varphi_2, f)
\end{bmatrix},
\]
gives

\[
\begin{bmatrix}
1 & 1/2 & 1/3 \\
1/2 & 1/3 & 1/4 \\
1/3 & 1/4 & 1/5 \\
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
c_2 \\
\end{bmatrix}
= 
\begin{bmatrix}
1.7183 \\
1.0000 \\
0.7183 \\
\end{bmatrix}
\]
We solve this system and obtain

\[ c_0 = 1.0130, \; c_1 = 0.8511, \; \text{and} \; c_2 = 0.8392. \]

Thus, the least squares polynomial of degree 2 fitting the proceeding data is

\[ p_2(x) = c_0 + c_1x + c_2x^2 = 1.0130 + 0.8511x + 0.8392x^2, \]

and the total error

\[ Q = Q(c_0, c_1, c_2) = \int_a^b [f(x) - p_2(x)]^2 \, dx = 2.7835 \times 10^{-5}. \]
Contents

1. Interpolating Polynomials
   - Existence and Uniqueness of Interpolating Polynomials

2. Lagrange Interpolating
   - Lagrange Interpolating Polynomials
   - Lagrange Error Terms

3. Newton Interpolation
   - Divided Differences
   - Newton's interpolating polynomial

4. Hermite Interpolation

5. Piecewise Polynomial Interpolation
   - Runge's Phenomenon
   - Piecewise Linear Interpolation
   - Piecewise Cubic Hermite Interpolation

6. Cubic Spline Interpolation
Orthogonal Polynomials

Definition

\( \varphi_0, \varphi_1, \ldots, \varphi_n \) is said to be an **orthogonal set of functions** for the interval \([a, b]\) with respect to the weight function \( \rho \) if

\[
(\varphi_j, \varphi_k) = \int_a^b \rho(x)\varphi_j(x)\varphi_k(x)dx = \left\{ \begin{array}{ll}
0, & \text{when } j \neq k, \\
\alpha_k > 0, & \text{when } j = k.
\end{array} \right.
\]

(5.10.1)

If, in addition, \( \alpha_k = 1 \) for each \( k = 0, 1, \ldots, n \), the set is said to be **orthogonal**.
Theorem

Suppose \( f \in C[a, b] \). If \( \{\varphi_0, \varphi_1, \cdots, \varphi_n\} \) is an orthogonal set of functions on an interval \([a, b]\) with respect to the weight function \( \rho \), then the least squares approximation to \( f \) on \([a, b]\) with respect to \( \rho \) is

\[
p_n(x) = \sum_{k=0}^{n} c_k x^k(x),
\]

where, for each \( k = 0, 1, \cdots, n \),
Interpolating Polynomials
Lagrange Interpolating
Newton Interpolation
Hermite Interpolation
Piecewise Polynomial Interpolation
Cubic Spline Interpolation
Least Squares Method
Least Squares Approximation
Orthogonal Polynomials

Basic Concepts

\[ c_k = \frac{\langle \varphi_k, f \rangle}{\varphi_k, \varphi_k} = \frac{\int_a^b \rho(x) \varphi_k(x) f(x) \, dx}{\int_a^b \rho(x) [\varphi_k(x)]^2 \, dx} = \frac{1}{\alpha_k} \int_a^b \rho(x) \varphi_k(x) f(x) \, dx. \]

(5.10.3)

Proof

From the discussion in Section 5.9, the least squares approximation to \( f \) on \([a, b]\) with respect to \( \rho \) is

\[ p_n(x) = \sum_{k=0}^{n} c_k x^k(x), \]

where the coefficients \( c_0, c_1, \ldots, c_n \) are determined by the normal system of equations in (5.9.6). Since \( \rho \) is positive on \([a, b]\), the system is well-defined.

Fan Yang: yfggd114@163.com  Interpolation
where the coefficients $c_0, c_1, \cdots, c_n$ are determined by
the normal system of equations in (5.9.6). Since $\{\varphi_0, \varphi_1, \cdots, \varphi_n\}$
is an orthogonal set of functions on the interval $[a, b]$ with
respect to the weight function $\rho$, Definition 5.10.1 gives

$$(\varphi_j, \varphi_k) = \int_a^b \rho(x)\varphi_j(x)\varphi_k(x)dx = \begin{cases} 0, & \text{when } j \neq k, \\ \alpha_k > 0, & \text{when } j = k, \end{cases}$$

where $\alpha_k = (\varphi_k, \varphi_k) = \int_a^b \rho(x)[\varphi_k(x)]^2dx$. So (5.9.6) now
becomes
which implies that

which implies that
\[ c_k = \frac{(\varphi_k, f)}{\varphi_k, \varphi_k} = \frac{\int_a^b \rho(x) \varphi_k(x) f(x) \, dx}{\int_a^b \rho(x) [\varphi_k(x)]^2 \, dx} = \frac{1}{\alpha_k} \int_a^b \rho(x) \varphi_k(x) f(x) \, dx. \]

for each \( k = 0, 1, \cdots, n \).

**Note**

If \( \varphi_0, \varphi_1, \cdots, \varphi_n \) is an orthogonal set of functions on an interval \([a, b]\) with respect to the weight function \( \rho \), the least squares approximation problem is greatly simplified: we can obtain the least squares polynomial to \( f \) directly from (5.10.3) without solving a linear system defined in (5.9.6).
Theorem

Suppose $\varphi_n$ is a polynomial of degree $n$, for each $n = 1, 2, \ldots$. Then the set of polynomial functions $\{\varphi_n\}$ is orthogonal on $[a, b]$ with respect to the weight function $\rho$ if and only if for any polynomial $Q_k$ of degree $k$,

$$
(\varphi_n, Q_k) = \int_a^b \rho(x)\varphi_n(x)Q_k(x)dx = 0 \quad (5.10.4)
$$

holds for each $n > k$. 
Proof

Suppose \( \{\varphi_n\} \) is orthogonal on \([a, b]\), where \( \varphi_n \) is a polynomial of degree \( n \). The theory in Linear Algebra implies that \( \{\varphi_n\} \) is a linearly independent set. Let \( Q_k \) be a polynomial of degree \( k \). Then there exist numbers \( c_0, c_1, \ldots, c_k \) such that

\[
Q_k(x) = \sum_{j=0}^{k} c_j \varphi_j(x).
\]
Thus,

\[(\varphi_n, Q_k) = \int_a^b \rho(x)\varphi_n(x)Q_k(x)dx = \sum_{j=0}^k c_j \int_a^b \rho(x)\varphi_n(x)\varphi_j(x)dx\]

\[= \sum_{j=0}^k c_j(\varphi_n, \varphi_j) = \sum_{j=0}^k c_j \times 0 = 0,\]

since \(\varphi_n\) is orthogonal to \(\varphi_j\) for each \(j = 0, 1, \ldots, k\).
Conversely, suppose, for any polynomial $Q_k$ of degree $k$, (5.10.4) holds for each $n > k$. When $m \neq n$, (5.10.4) gives

$$
(\varphi_m, \varphi_n) = \int_a^b \rho(x) \varphi_m(x) \varphi_n(x) \, dx = 0. \quad (5.10.5)
$$

When $m = n$,

$$
(\varphi_m, \varphi_n) = \int_a^b \rho(x)[\varphi_n(x)]^2 \, dx > 0, \quad (5.10.6)
$$

provided that $\varphi_n(x)$ is not a zero polynomial.
Combining (5.10.5) and (5.10.6) yields that the set of polynomial functions \( \{ \varphi_n(x) \} \) is orthogonal on \([a, b]\) with respect to the weight function \( \rho \) from Definition 5.10.1.