Chapter 4

Polynomial Interpolation

In this chapter, we consider the important problem of approximating a function $f(x)$, whose values at a set of distinct points $x_0, x_1, x_2, \ldots, x_n$ are known, by a polynomial $P(x)$ such that $P(x_i) = f(x_i)$, $i = 0, 1, 2, \ldots, n$. Such a polynomial is known as an interpolating polynomial. An approximation of this function is desirable if $f(x)$ is difficult to evaluate or manipulate like the error function

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt,$$

for example. Note that polynomials are easily evaluated, differentiated or integrated. Another purpose of such an approximation is that a very long table of function values $f(x_i)$ may be replaced by a short table and a compact interpolation subroutine.

4.1 Linear Interpolation

Let $f(x)$ be given at two distinct points $x_i$ and $x_{i+1}$. Now, given $f(x_i) = f_i$ and $f(x_{i+1}) = f_{i+1}$, we wish to determine the interpolating polynomial $P_1(x)$ of degree 1 (i.e., a straight line) that approximates $f(x)$ on the interval $[x_i, x_{i+1}]$.

Let $P_1(x) = \alpha x + \beta$. From the conditions $P_1(x_i) = f_i$ and $P_1(x_{i+1}) = f_{i+1}$ we obtain the two equations

$$\begin{align*}
\alpha x_i + \beta &= f_i \\
\alpha x_{i+1} + \beta &= f_{i+1},
\end{align*}$$

i.e.,

$$\begin{pmatrix} x_i & 1 \\ x_{i+1} & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} f_i \\ f_{i+1} \end{pmatrix}.$$ 

This system is clearly nonsingular since $x_i \neq x_{i+1}$, and it has a unique solution:

$$\begin{align*}
\alpha &= \frac{f_{i+1} - f_i}{x_{i+1} - x_i} \\
\beta &= \frac{f_i x_{i+1} - f_{i+1} x_i}{x_{i+1} - x_i}.
\end{align*}$$

Therefore,

$$P_1(x) = f_i + \left( \frac{f_{i+1} - f_i}{x_{i+1} - x_i} \right) (x - x_i).$$

Example 4.1 Using linear interpolation, approximate $\sin 6.5^\circ$ from a table which gives

(a) $\sin 6^\circ = 0.10453$; $\sin 7^\circ = 0.12187$.

(b) $\sin 0^\circ = 0$; $\sin 10^\circ = 0.17365$. 

Solution

(a) \( P_1(6.5) = 0.10453 + \left( \frac{0.12187 - 0.10453}{7 - 6} \right) \approx 0.11320. \)

(b) \( P_1(6.5) = 0 + \frac{0.17365}{10}(6.5) \approx 0.11287. \)

The first answer is correct to 5 decimals whereas the second answer is correct only to 2 decimals!  

Conclusion: Linear interpolation is suitable only over small intervals.

4.2 Polynomial Interpolation

Since linear interpolation is not adequate unless the given points are closely spaced, we consider higher order interpolating polynomials. Let \( f(x) \) be given at the selected sample of \( (n + 1) \) points: \( x_0 < x_1 < \cdots < x_n \), i.e., we have \( (n + 1) \) pairs \( (x_i, f_i) \), \( i = 0, 1, 2, \ldots, n \). The objective now is to find the lowest degree polynomial that passes through this selected sample of points.

Consider the \( n \)th degree polynomial

\[
P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n.
\]

We wish to determine the coefficients \( a_j, \ j = 0, 1, \ldots, n, \) such that

\[
P_n(x_j) = f(x_j), \quad j = 0, 1, 2, \ldots, n.
\]

These \( (n + 1) \) conditions yield the linear system

\[
\begin{align*}
0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= f_0 \\
0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= f_1 \\
0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_n x_2^n &= f_2 \\
\vdots & \vdots \ \vdots & \vdots \\
0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= f_n
\end{align*}
\]

or \( Va = f \), where \( a^T = (a_0, a_1, \ldots, a_n) \), \( f^T = (f_0, f_1, \ldots, f_n) \), and \( V \) is an \( (n + 1) \times (n + 1) \) matrix given by

\[
V = \begin{pmatrix}
1 & x_0 & x_0^2 & \cdots & x_0^n \\
1 & x_1 & x_1^2 & \cdots & x_1^n \\
1 & x_2 & x_2^2 & \cdots & x_2^n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_n & x_n^2 & \cdots & x_n^n
\end{pmatrix}
\]
The matrix $V$ is known as the Vandermonde matrix. It is nonsingular with nonzero determinant

$$\det(V) = \prod_{i=0}^{n-1} (x_i - x_j).$$

For $n = 3$, for example,

$$\det(V) = (x_1 - x_0)(x_2 - x_0)(x_3 - x_0)(x_2 - x_1)(x_3 - x_1)(x_3 - x_2)$$

Hence $a$ can be uniquely determined as $a = V^{-1}f$. Another proof of the uniqueness of the interpolating polynomial can be given as follows.

**Theorem 4.1 Uniqueness of interpolating polynomial.** Given a set of points $x_0 < x_1 < \ldots < x_n$, there exists only one polynomial that interpolates a function at those points.

**Proof** Let $P(x)$ and $Q(x)$ be two interpolating polynomials of degree at most $n$, for the same set of points $x_0 < x_1 < \ldots < x_n$. Also, let

$$R(x) = P(x) - Q(x).$$

Then $R(x)$ is also a polynomial of degree at most $n$. Since

$$P(x_i) = Q(x_i) = f_i,$$

we have, $R(x_i) = 0$ for $i = 0, 1, \ldots, n$. In other words $R(x)$ has $(n + 1)$ roots. From the Fundamental Theorem of Algebra however, $R(x)$ cannot have more than $n$ roots. A contradiction! Thus, $R(x) \equiv 0$, and $P(x) \equiv Q(x)$. \qed

**Example 4.2** Given the following table for the function $f(x)$, obtain the lowest degree interpolating polynomial.

<table>
<thead>
<tr>
<th>$x_k$</th>
<th>-1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_k$</td>
<td>-1</td>
<td>-4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Solution** Since the number of nodes $= 3 = n + 1$, therefore $n = 2$. Let

$$P_2(x) = a_0 + a_1 x + a_2 x^2,$$

that satisfies the following conditions at the points $x_0 = 1; x_1 = 2, x_2 = 4$:

$$P(-1) = -1; \quad P(2) = -4; \quad P(4) = 4.$$ 

$$\begin{pmatrix} 1 & -1 & 1 \\ 1 & 2 & 4 \\ 1 & 4 & 16 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -4 \\ 4 \end{pmatrix}.$$ 

Using Gaussian elimination with partial pivoting, we compute the solution

$$a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} -4 \\ -2 \\ 1 \end{pmatrix},$$

Therefore, the interpolating polynomial is given by

$$P_2(x) = -4 - 2x + x^2.$$ 

We can use this approximation to estimate the root of $f(x)$ that lies in the interval $[2, 4]$:

$$\gamma = \frac{2 + \sqrt{4 + 16}}{2} = 1 + \sqrt{5}.$$
An important remark is in order. One, in general, should not determine the interpolating polynomial by solving the Vandermonde linear system. These systems are surprisingly ill-conditioned for \( n \) no larger than 10. For example, for \( 0 < x_0 < x_1 < \cdots < x_n = 1 \) uniformly distributed in \([0,1]\), large \( n \) yields a Vandermonde matrix with almost linearly dependent columns, and the Vandermonde system becomes almost singular.

A more satisfactory form of the interpolating polynomial is due to Lagrange,

\[
P_n(x) = f(x_0)L_0(x) + f(x_1)L_1(x) + \cdots + f(x_n)L_n(x).
\]

Here \( L_j(x) \), for \( 0 \leq j \leq n \), are polynomials of degree \( n \) called fundamental polynomials or Lagrange polynomials. From the \((n+1)\) conditions

\[
P_n(x_i) = f(x_i), \quad 0 \leq i \leq n,
\]

we see that

\[
L_j(x_i) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \\ \end{cases}
\]

The above property is certainly satisfied if we choose

\[
L_j(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n)}{(x_j - x_0)(x_j - x_1)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)}
\]

i.e.,

\[
L_j(x) = \prod_{i=0, i \neq j}^{n} \frac{x - x_i}{x_j - x_i}.
\]

The fact that \( L_j(x) \) is unique follows from the uniqueness of the interpolation polynomial.

Recall that there is one and only one polynomial of degree \( n \) or less that interpolates \( f(x) \) at the \((n+1)\) nodes \( x_0 < x_1 < \cdots < x_n \). Thus, for this given set of nodes, both forms yield the same polynomial!

**Vandermonde Approach.**

\[
P_n(x) = \sum_{i=0}^{n} a_i x^i \quad V a = f
\]

Operation count: It can be shown that the Vandermonde system \( V a = f \) can be solved in \( O(n^2) \) arithmetic operations rather than \( O(n^3) \) operations because of the special form of the Vandermonde matrix.

**Lagrange Approach.**

\[
P_n(x) = \sum_{j=0}^{n} f_j L_j(x) = \sum_{j=0}^{n} g_j \left[ \prod_{i=0, i \neq j}^{n} \frac{x - x_i}{x_j - x_i} \right] = \sum_{j=0}^{n} g_j \cdot v_j
\]

where

\[
g_j = \frac{f(x_j)}{\prod_{i=0, i \neq j}^{n} (x_j - x_i)}.
\]

Operation count: The Lagrange form avoids solving the ill-conditioned Vandermonde linear, systems, and evaluates the polynomials

\[
v_j = \frac{1}{(x - x_j)} \prod_{i=0}^{n} (x - x_i)
\]
for a given \( x \). The cost of these operations for each \( j = 0, 1, 2, \ldots, n \) are:

\[
\begin{align*}
  u(x) &= \prod_{i=0}^{n}(x - x_i) \quad \rightarrow (2n + 1) \text{ ops} \\
  \nu_j &= \frac{u(x)}{(x - x_j)} f_j \quad \rightarrow (n + 1) \text{ ops} \\
  g_j &= \prod_{i=0, i \neq j}^{n}(x_j - x_i) \quad \rightarrow 2n(n + 1) \text{ ops (evaluated once only)}
\end{align*}
\]

Now, the cost for evaluating the polynomial

\[
P_n(x) = \sum_{j=0}^{n} g_j \nu_j \quad \rightarrow (2n + 1) \text{ ops}
\]

Therefore, the total number of operations are \( 5n + 3 \) once \( g_j \)'s are evaluated.

### 4.3 Error in Polynomial Interpolation

Let \( e_n(x) \) be the error in polynomial interpolation given by

\[
e_n(x) = f(x) - P_n(x).
\]

Since \( P_n(x_i) = f(x_i) \), we have

\[
e_n(x_i) = 0 \quad x_0 < x_1 < \cdots < x_n.
\]

In other words \( e_n(x) \) is a function with at least \((n + 1)\) roots. Hence, we can write

\[f(x)\]

\[P_n(x)\]

\[x_0 \quad x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_n\]

Figure 4.2: Computing the error in polynomial interpolation.

\[
e_n(x) = (x - x_0)(x - x_1)\cdots(x - x_n)h(x) = \pi(x) \cdot h(x)
\]

where \( h(x) \) is a function of \( x \). Let us define

\[
g(z) = e_n(z) - \pi(z)h(x)
\]

where \( x \neq x_j \), for \( 0 \leq j \leq n \). Thus,

\[
\begin{align*}
  g(x) &= 0, \\
  g(x_j) &= 0, \quad 0 \leq j \leq n,
\end{align*}
\]
i.e., \( g(z) \) has \((n+2)\) distinct roots. Assuming that \( f(x) \) has at least \( n+1 \) continuous derivatives, the \((n+1)\)th derivative of \( g(z) \), i.e.,

\[
\frac{d^{(n+1)}}{dz^{(n+1)}} g(z) \equiv g^{(n+1)}(z)
\]

has at least one root \( x^* \). Hence,

\[
g^{(n+1)}(z) = e^{(n+1)}_n(z) - h(x)\pi^{(n+1)}(z),
\]

where \( \pi^{(n+1)}(z) = (n+1)! \). Moreover, since \( e_n(z) = f(z) - P_n(z) \), therefore,

\[
e^{(n+1)}_n(z) = f^{(n+1)}(z) - P^{(n+1)}_n(z)
\]

in which the last term \( P^{(n+1)}_n(z) \) is zero. Using the above equations, we have

\[
g^{(n+1)}(z) = f^{(n+1)}(z) - h(x)(n+1)!
\]

We had shown earlier that \( g^{(n+1)}(z) \) has at least one root \( x^* \) in the interval containing \( x \) and \( x_j, j = 0, \ldots, n \). This implies that \( g^{(n+1)}(x^*) = 0 \), and therefore,

\[
h(x) = \frac{f^{(n+1)}(x^*)}{(n+1)!}
\]

Consequently, we have the following representation of the error in polynomial interpolation

\[
e_n(x) = f(x) - P_n(x) = (x - x_0)(x - x_1)\cdots(x - x_n)\frac{f^{(n+1)}(x^*)}{(n+1)!}
\]

where \( x^* \) is in the interval containing \( x \) and \( x_j, j = 0, \ldots, n \).
Example 4.3 Obtain the Lagrange interpolating polynomial from the data

\[
\begin{align*}
\sin 0 &= 0, & \sin \frac{\pi}{6} &= \frac{1}{2}, & \sin \frac{\pi}{3} &= \frac{\sqrt{3}}{2}, & \sin \frac{\pi}{2} &= 1,
\end{align*}
\]

and use it to evaluate \(\sin \frac{\pi}{12}\) and \(\sin \frac{\pi}{4}\).

Solution The interpolating polynomial is given by

\[
P_3(x) = f_0L_0(x) + f_1L_1(x) + f_2L_2(x) + f_3L_3(x)
\]

At \(x = \frac{\pi}{12}\),

\[
\begin{align*}
L_1 \left( \frac{\pi}{12} \right) &= \frac{\left( \frac{\pi}{12} - 0 \right) \left( \frac{\pi}{12} - \frac{\pi}{6} \right) \left( \frac{\pi}{12} - \frac{\pi}{3} \right)}{\left( \frac{\pi}{6} - 0 \right) \left( \frac{\pi}{6} - \frac{\pi}{3} \right) \left( \frac{\pi}{6} - \frac{\pi}{2} \right)} = \frac{15}{16} \\
L_2 \left( \frac{\pi}{12} \right) &= \frac{\left( \frac{\pi}{12} - 0 \right) \left( \frac{\pi}{12} - \frac{\pi}{6} \right) \left( \frac{\pi}{12} - \frac{\pi}{3} \right)}{\left( \frac{\pi}{4} - 0 \right) \left( \frac{\pi}{4} - \frac{\pi}{6} \right) \left( \frac{\pi}{4} - \frac{\pi}{2} \right)} = \frac{5}{16} \\
L_3 \left( \frac{\pi}{12} \right) &= \frac{\left( \frac{\pi}{12} - 0 \right) \left( \frac{\pi}{12} - \frac{\pi}{6} \right) \left( \frac{\pi}{12} - \frac{\pi}{3} \right)}{\left( \frac{\pi}{2} - 0 \right) \left( \frac{\pi}{2} - \frac{\pi}{6} \right) \left( \frac{\pi}{2} - \frac{\pi}{3} \right)} = \frac{1}{16}
\end{align*}
\]

Therefore,

\[
P_3 \left( \frac{\pi}{12} \right) = 0.26062.
\]

Since \(\sin \frac{\pi}{12} = 0.25882\), we have \(|\text{error}| = 0.0018\).

At \(x = \frac{\pi}{4}\),

\[
\begin{align*}
L_1 \left( \frac{\pi}{4} \right) &= 0.5625, & L_2 \left( \frac{\pi}{4} \right) &= 0.5625, & L_3 \left( \frac{\pi}{4} \right) &= 0.0625.
\end{align*}
\]

Therefore

\[
P_3 \left( \frac{\pi}{4} \right) = 0.70589.
\]

Since \(\sin \frac{\pi}{4} = 0.70711\), we have \(|\text{error}| = 0.00122\).

Let us estimate the error \(e_n(x)\) at \(x = \frac{\pi}{4}\):

\[
e_3 \left( \frac{\pi}{4} \right) = \frac{\left( \frac{\pi}{4} - 0 \right) \left( \frac{\pi}{4} - \frac{\pi}{6} \right) \left( \frac{\pi}{4} - \frac{\pi}{3} \right) \left( \frac{\pi}{2} - \frac{\pi}{6} \right) \left( \frac{\pi}{2} - \frac{\pi}{3} \right) \left( \frac{\pi}{2} - \frac{\pi}{4} \right)}{4!} f^{(4)}(x)\]
\]

Since \(f(x) = \sin x\); \(f^{(4)}(x) = \sin x\), and thus,

\[
|f^{(4)}(x)| \leq 1.
\]

Therefore,

\[
|e_3 \left( \frac{\pi}{4} \right)| \leq \frac{\pi}{4} \cdot \frac{\pi}{12} \cdot \frac{\pi}{4} \cdot 1 \cdot \frac{1}{4!} = 1.76 \times 10^{-3}.
\]

Similarly, \(e_n(x)\) at \(x = \frac{\pi}{12}\) is bounded by

\[
|e_3 \left( \frac{\pi}{12} \right)| \leq \frac{\pi}{12} \cdot \frac{\pi}{4} \cdot \frac{5\pi}{12} \cdot \frac{1}{4!} = 2.94 \times 10^{-3}.
\]

\(\square\)
4.4 Runge’s Function and Equidistant Interpolation

In this section, we discuss a problem that arises when one constructs a global interpolation polynomial on a fairly large number of equally spaced nodes. Consider the problem of approximating the following function on the interval $[-1, 1]$

$$f(x) = \frac{1}{1 + 25x^2}$$

using the interpolation polynomial $P_n(x)$ on the equidistant nodes

$$x_j = -1 + \frac{2j}{n}, \quad j = 0, 1, \ldots, n.$$

Figure 4.4 shows how the polynomials $P_4(x), P_8(x), P_{12}(x),$ and $P_{16}(x)$ approximate $f(x)$ on $[-1, 1]$. We see that as $n$ increases, the interpolation error in the central portion of the interval diminishes, while increasing rapidly near the ends of the interval. Such behavior is typical in equidistant interpolation with polynomials of high degree. This is known as Runge’s phenomenon, and the function $f(x)$ is known as Runge’s function.

If the $(n + 1)$ interpolation nodes are not chosen equally spaced, but rather placed near the ends of the interval at the roots of the so-called Chebyshev polynomial of degree $(n + 1)$, the problem with Runge’s function disappears (see Fig. 4.5). The roots of the Chebyshev polynomial of degree $(n + 1)$ are given by

$$x_k = -\cos \left( \frac{k + \frac{1}{2}}{n + 1} \pi \right), \quad k = 0, 1, \ldots, n.$$
4.5 The Newton Representation

The Lagrange form for the interpolating polynomial is not suitable for practical computations. The Newton form of the interpolating polynomial proves to be much more convenient. Let us obtain a different form of the lowest degree interpolating polynomial at the distinct interpolation nodes given by

\[ x_0 < x_1 < x_2 < \cdots < x_{n-1} \]

where \( f(x_j) = f_j, \quad j = 0, 1, \ldots, n - 1 \). Let the interpolating polynomial be

\[
P_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}).
\]

Using the \( n \) conditions

\[ P_n(x_j) = f_j, \quad j = 0, 1, \ldots, n - 1, \]

we can construct a system of equations in the unknown \( c_j \). For \( n = 3 \), for example, this yields a system of linear equations of the form \( Lc = f \), where \( L \) is given by,

\[
L = \begin{pmatrix}
1 & 0 & 0 & 0 \\
1 & (x_1 - x_0) & 0 & 0 \\
1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & 0 \\
1 & (x_3 - x_0) & (x_3 - x_0)(x_3 - x_1) & (x_3 - x_0)(x_3 - x_1)(x_3 - x_2)
\end{pmatrix}.
\]
Note that the evaluation of \( L \) for \( n \) nodes requires \( O(n^2) \) arithmetic operations. Assuming that the nodes are equidistant, i.e., \( x_{i+1} - x_i = h \), the system \( L \mathbf{c} = \mathbf{f} \) is given by

\[
\begin{pmatrix}
1 & 0 & 0 \\
1 & h & 0 \\
1 & 2h & 2h^2 \\
1 & 3h & 6h^2 & 6h^3
\end{pmatrix}
\begin{pmatrix}
c_0 \\
c_1 \\
c_2 \\
c_3
\end{pmatrix}
= \begin{pmatrix}
f_0 \\
f_1 \\
f_2 \\
f_3
\end{pmatrix}.
\]

**Step 1.**

\[
c_0 = f_0 \\
f_1^{(1)} = (f_1 - f_0)/h \\
f_2^{(1)} = (f_2 - f_0)/2h \\
f_3^{(1)} = (f_3 - f_0)/3h.
\]

Then we have the system order 3:

\[
\begin{pmatrix}
1 & 0 & 0 \\
1 & h & 0 \\
1 & 2h & 2h^2 \\
1 & 3h & 6h^2 & 6h^3
\end{pmatrix}
\begin{pmatrix}
c_1 \\
c_2 \\
c_3
\end{pmatrix}
= \begin{pmatrix}
f_1^{(1)} \\
f_2^{(1)} \\
f_3^{(1)}
\end{pmatrix}.
\]

**Step 2.**

\[
c_1 = f_1^{(1)} \\
f_2^{(2)} = (f_2^{(1)} - f_1^{(1)})/h \\
f_3^{(2)} = (f_3^{(1)} - f_1^{(1)})/2h.
\]

Then we get the system of order 2:

\[
\begin{pmatrix}
1 & 0 \\
1 & h
\end{pmatrix}
\begin{pmatrix}
c_2 \\
c_3
\end{pmatrix}
= \begin{pmatrix}
f_2^{(2)} \\
f_3^{(2)}
\end{pmatrix}.
\]

**Step 3.**

\[
c_2 = f_2^{(2)} \\
f_3^{(3)} = (f_3^{(2)} - f_2^{(2)})/h.
\]

**Step 4.**

\[
c_3 = f_3^{(3)}
\]

**MATLAB**

```matlab
function c = InterpolateNewto(x,f)
    n = length(x);
    for k = 1:n-1
        f(k+1:n) = (f(k+1:n)-f(k))./(x(k+1:n)-x(k));
    end
    c = y;
```
4.6 Spline Interpolation

Draftsmen used to draw smooth curves through data points by using splines. These are thin flexible strips of plastic or wood which were laid on paper and held with weights so as to pass through the required data points (or nodes). The weights are constructed in such a way that the spline is free to slip. As a result, the flexible spline straightens out as much as it can subject to passing over these points. Theory of elasticity suggests that this mechanical spline is given by a cubic polynomial (degree 3 polynomial) in each subinterval $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, n − 1$, with adjacent cubics joint continuously with continuous first and second derivatives. Let

$$s(x) = a_0 + a_1x + a_2x^2 + a_3x^3,$$

be the form of each cubic spline. Since we have $(n − 1)$ subintervals $[x_i, x_{i+1}]$, $1 \leq i \leq n − 1$, then we have $4(n − 1)$ unknowns; (4 parameters for each subinterval: $x_0, a_1, a_2, a_3$). Suppose $s_i(x)$ denotes the cubic spline the $i$th interval. The conditions to be satisfied by these cubic splines are:

1. Continuity at each interior node:

$$s_{i−1}(x_i) = s_i(x_i), \quad i = 2, 3, \ldots, n − 1.$$

2. Continuity of first derivative at each interior node:

$$s'_{i−1}(x_i) = s'_i(x_i), \quad i = 2, 3, \ldots, n − 1.$$

3. Continuity of second derivative at each interior node:

$$s''_{i−1}(x_i) = s''_i(x_i), \quad i = 2, 3, \ldots, n − 1.$$

4. Interpolation of function at each node:

$$s(x_j) = f(x_j), \quad j = 1, 2, \ldots, n.$$

These provide $3(n − 2) + n = 4n − 6$ conditions; however, in order to completely specify the cubic splines, we still need 2 additional conditions. These are conditions specified at the end points $x_1$ and $x_n$. For the “mechanical” or “natural” spline we set

$$s''(x_1) = s''(x_n) = 0.$$
Let the nodes $x_j$, $j = 1, 2, \ldots, n$, be equidistant, i.e.,

$$x_{j+1} - x_j = h$$

and let,

$$\sigma_j = \frac{1}{6} s''(x_j)$$

i.e.,

$$s''_{i-1}(x_i) = s''(x_i) = 6\sigma_i$$

for all $i$ satisfying condition ??. Since

$$s(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3,$$

we have

$$s'(x) = a_1 + 2a_2 x + 3a_3 x^2$$

$$s''(x) = 2a_2 + 6a_3 x,$$

i.e., $s''(x)$ is a straight line. Thus, $s''(x)$ takes the following form (see Fig. ??):

$$s''_i(x) = \frac{x_{i+1} - x}{h}(6\sigma_i) + \frac{(x - x_i)}{h}(6\sigma_{i+1}), \quad i = 1, 2, \ldots, n - 1.$$

![Figure 4.7: Computing spline within an interval.](image)

Integrating once, we have

$$s'_i(x) = -\frac{6\sigma_i}{h} \cdot \frac{(x_{i+1} - x)^2}{2} + \frac{6\sigma_{i+1}}{h} \cdot \frac{(x - x_i)^2}{2} + \beta_1$$

and integrating once more, we have

$$s_i(x) = \frac{\sigma_i}{h}(x_{i+1} - x)^3 + \frac{\sigma_{i+1}}{h}(x - x_i)^3 + \beta_1 x + \beta_2$$

where $\beta_1$ and $\beta_2$ are constants. Clearly we can write $s_i(x)$ as follows,

$$s_i(x) = \frac{\sigma_i}{h}(x_{i+1} - x)^3 + \frac{\sigma_{i+1}}{h}(x - x_i)^3 + \gamma_1(x - x_i) + \gamma_2(x_{i+1} - x).$$

Using conditions ?? and ??,

$$s_i(x_i) = f_i \quad \Rightarrow \quad f_i = \sigma_i h^2 + \gamma_2 h$$

$$s_i(x_{i+1}) = f_{i+1} \quad \Rightarrow \quad f_{i+1} = \sigma_{i+1} h^2 + \gamma_1 h$$
Therefore,
\[ \gamma_1 = \frac{f_{i+1}}{h} - \sigma_{i+1}h, \]
\[ \gamma_2 = \frac{f_i}{h} - \sigma_i h, \]
and
\[ s_i(x) = \frac{\sigma_i}{h} (x_{i+1} - x)^3 + \frac{\sigma_{i+1}}{h} (x-x_i)^3 + \left( \frac{f_{i+1}}{h} - \sigma_{i+1}h \right) (x-x_i) \]
\[ + \left( \frac{f_i}{h} - \sigma_i h \right) (x_{i+1} - x) \]
\[ s'_i(x) = \frac{-3\sigma_i}{h} (x_{i+1} - x)^2 + \frac{3\sigma_{i+1}}{h} (x-x_i)^2 + \left( \frac{f_{i+1}}{h} - \sigma_{i+1}h \right) \]
\[ - \left( \frac{f_i}{h} - \sigma_i h \right) . \]
Using condition ??, i.e., \( s'_i(x_i) = s'_{i-1}(x_i) \), we get
\[ -3\sigma_i h + \left( \frac{f_{i+1} - f_i}{h} \right) - h(\sigma_{i+1} - \sigma_i) = 3\sigma_i h + \left( \frac{f_i - f_{i-1}}{h} \right) - h(\sigma_i - \sigma_{i-1}) \]
or
\[ \sigma_{i+1} + 4\sigma_i + \sigma_{i-1} = \frac{\Delta_i - \Delta_{i-1}}{h} \quad i = 2, 3, \ldots, n-1, \]
where
\[ \Delta_i = \frac{f_{i+1} - f_i}{h}. \]
Using the additional conditions \( \sigma_1 = \sigma_n = 0 \), we have
\[ 4\sigma_2 + \sigma_3 = \frac{(\Delta_2 - \Delta_1)}{h} \]
\[ \sigma_{i-1} + 4\sigma_i + \sigma_{i+1} = \frac{(\Delta_i - \Delta_{i-1})}{h} \quad i = 3, 4, \ldots, n-2 \]
\[ \sigma_{n-2} + 4\sigma_{n-1} = \frac{(\Delta_{n-1} - \Delta_{n-2})}{h}. \]
Now, assigning
\[ g_j = \frac{\Delta_j - \Delta_{j-1}}{h} = \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2}, \]
we have the linear system
\[ \begin{pmatrix} 4 & 1 \\ 1 & 4 & 1 \\ \vdots & \vdots & \vdots \\ 1 & 4 & 1 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} \sigma_2 \\ \sigma_3 \\ \vdots \\ \sigma_{n-2} \\ \sigma_{n-1} \end{pmatrix} = \begin{pmatrix} g_2 \\ g_3 \\ \vdots \\ g_{n-2} \\ g_{n-1} \end{pmatrix} . \]
The tridiagonal system above, denoted by \([1, 4, 1]\), may be factored as follows:
\[ \begin{pmatrix} 1 & \lambda_1 & 1 \\ \lambda_2 & 1 & \lambda_3 \\ \vdots & \vdots & \ddots \\ \lambda_{n-3} & 1 & \lambda_{n-2} \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \vdots \\ \mu_{n-2} \end{pmatrix} \]
The following conditions must be satisfied:
\[ \mu_1 = 4, \quad \lambda_i \mu_i = 1, \quad \lambda_i + \mu_{i+1} = 4 \]
The algorithm for computing \( L \) and \( U \) is given below.
Triangular decomposition for system [1, 4, 1]

\[ \mu_1 = 4 \]

for \( i = 1, 2, \ldots, n - 3 \)

\[ \lambda_i = \mu_{i-1} \]

\[ \mu_{i+1} = 4 - \lambda_i \]

end;

Once \( L \) and \( U \) are obtained, the above system is solved as shown in Chapter ??.