Review  We have constructed the Newton formula for Lagrange interpolation:

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

is the degree-\( n \) polynomial \( p_n \) satisfying \( p_n(x_j) = f_j \) for \( 0 \leq j \leq n \). Here \( f[x_j] = f(x_j) \) and

\[ f[x_0, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}. \]

(A consequence is the symmetry of divided differences: every divided difference is independent of the ordering of the interpolation points.) The error satisfies

\[ f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \omega(x) \]

where \( \xi \) is some unknown point contained in the interval \([\min x_j, \max x_j]\) and

\[ \omega(x) = (x - x_0)(x - x_1) \cdots (x - x_n). \]

Differences and derivatives  The Newton formula makes it simple to add \( x_{n+1} = x \) as an additional interpolation point, and the value \( p_{n+1}(x) = f(x) \) is then exact:

\[ f(x) = p_{n+1}(x) = p_n(x) + f[x_0, x_1, \ldots, x_{n-1}, x_n] \omega(x) \]

Comparing with the error formula proves a theorem: every divided difference is proportional to a derivative

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

evaluated at some unknown point \( \xi \) contained in the interval \([\min x_j, \max x_j]\). This proportionality is important because as the interpolation points \( x_j \) coalesce into a single point \( x \), they trap the unknown evaluation point \( \xi \) between them:

\[ f[x_0, x_1, \ldots, x_n] \to \frac{f^{(n+1)}(x)}{(n+1)!} \]

as all the \( x_j \)'s approach \( x \). So Newton’s formula has a limit as interpolation points coalesce. If the \( x_j \)'s all coalesce into a single point \( x \), Newton’s interpolation formula becomes a Taylor expansion for the Taylor polynomial which matches as many derivatives as it can with \( f \) at \( x \). For example, if \( n = 0 \) then the linear interpolant

\[ p(x) = f[x_0] + f[x_0, x_1](x - x_0) \]

has a limit as \( x_1 \to x_0 \) because

\[ f[x_0, x_1] \to f'(x_0) = f'_0 \]

as \( x_1 \to x_0 \). Thus

\[ p(x) = f[x_0] + f[x_0, x_0](x - x_0) = f(x_0) + f'(x_0)(x - x_0) \]

reproduces the first-order Taylor expansion.
**Hermite interpolation** bridges Lagrange interpolation and Taylor expansion by matching selected derivatives as well as function values at selected points. For example, let’s find a degree $2n + 1$ polynomial $p(x)$ with

$$p(x_j) = f_j$$

and

$$p'(x_j) = f'_j$$

for $0 \leq j \leq n$. Newton’s formula makes this easy. With two points we want

$$p(x_j) = f_j$$

and

$$p'(x_j) = f'_j.$$ 

We build up $p$ gradually

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + a_3(x - x_0)^2(x - x_1)$$

choosing each $a_j$ so that the previously satisfied interpolation conditions remain satisfied. To do this, build the difference table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0$</td>
<td>$f_0$</td>
<td>$a_1 = f'_0$</td>
<td>$a_2 = f[x_0, x_0, x_1]$</td>
<td>$a_3$</td>
<td>$a_4$</td>
<td>$a_5$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$f_1$</td>
<td>$f'[1]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Repeat $x$ values as many times as derivatives are to be specified. Substitute derivatives divided by factorials whenever divide by zero would otherwise occur.

For example, with $f(x) = 2^x$ so $f'(x) = (\log 2)2^x$ we build a cubic interpolant to $f$ and $f'$ at $x_0 = 0$ and $x_1 = 1$ with the table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$1$</td>
<td>$\log 2$</td>
<td>$1 - \log 2$</td>
<td>$3\log 2 - 2$</td>
</tr>
<tr>
<td>$0$</td>
<td>$1$</td>
<td>$1$</td>
<td>$2\log 2 - 1$</td>
<td></td>
</tr>
<tr>
<td>$1$</td>
<td>$2$</td>
<td>$2\log 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$1$</td>
<td>$2$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

so

$$p(x) = 1 + (\log 2)x + (1 - \log 2)x^2 + (3\log 2 - 2)x^2(x - 1)$$

is the Hermite interpolant in Newton form.

If we want to add a new interpolation point $x_2 = 2$, we just add two diagonals to the table and two terms to $p$:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$1$</td>
<td>$\log 2$</td>
<td>$1 - \log 2$</td>
<td>$3\log 2 - 2$</td>
<td>$(5 - 5\log 2)/2$</td>
<td>$(13\log 2 - 11)/4$</td>
</tr>
<tr>
<td>$0$</td>
<td>$1$</td>
<td>$1$</td>
<td>$2\log 2 - 1$</td>
<td>$3 - 2\log 2$</td>
<td>$(8\log 2 - 6)/2$</td>
<td></td>
</tr>
<tr>
<td>$1$</td>
<td>$2$</td>
<td>$2\log 2$</td>
<td>$2 - 2\log 2$</td>
<td>$6\log 2 - 3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$1$</td>
<td>$2$</td>
<td>$2$</td>
<td>$4\log 2 - 1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2$</td>
<td>$4$</td>
<td>$4\log 2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2$</td>
<td>$4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
and
\[ p(x) = 1 + (\log 2)x + (1 - \log 2)x^2 + (3 \log 2 - 2)x^2(x - 1) \\
+((5 - 5 \log 2)/2)x^2(x - 1)^2 + ((13 \log 2 - 11)/4)x^2(x - 1)^2(x - 2) \]
is the Hermite interpolant in Newton form.

Alternatively, we can match one more derivative \( f'' \) at \( x = 1 \) only:

\[
\begin{array}{cccccc}
  0 & 1 & \log 2 & 1 - \log 2 & 3 \log 2 - 2 & \log^2 2 - 5 \log 2 + 3 \\
  0 & 1 & 1 & 2 \log 2 - 2 \log 2 + 1 \\
  2 & 2 & \log 2 & \log^2 2 \\
  1 & 2 & 2 \log 2 \\
  1 & 2 \\
\end{array}
\]

Note that \( f[x_1, x_1, x_1] = f''(x_1)/2! \) contains a factor of 2! which is easy to forget about. The resulting interpolant is

\[ p(x) = 1 + (\log 2)x + (1 - \log 2)x^2 + (3 \log 2 - 2)x^2(x - 1) + (\log^2 2 - 5 \log 2 + 3)x^2(x - 1)^2. \]

**Error estimate** Suppose \( p \) matches \( f \) and \( f' \) at \( n + 1 \) points with degree \( 2n + 1 \). Inevitably the error must be

\[ f(x) - p(x) = Cf^{(2n+2)}(\xi)\omega(x)^2 \]

where the derivative ensures that polynomials of degree \( 2n + 1 \) are reproduced exactly (by uniqueness) and \( \omega(x)^2 \) ensures that \( p \) matches both \( f \) and \( f' \) at each \( x_j \). Determine \( C \) by testing the formula on \( f(x) = \omega(x)^2 \) where the derivative is \( (2n + 2)! \): then \( p(x) = 0 \) so

\[ f(x) - p(x) = \frac{1}{(2n + 2)!}f^{(2n+2)}(\xi)\omega(x)^2. \]

A subtle advantage of Hermite interpolation is that for a given spacing \( h = x_{j+1} - x_j \), the double roots of

\[ \omega(x)^2 = (x - x_0)^2(x - x_1)^2 \cdots (x - x_n)^2 \]

are twice as clustered as the single roots of the Lagrange equivalent of equal degree

\[ \omega_{2n}(x) = (x - x_0)(x - x_1) \cdots (x - x_{2n+1}), \]

Thus the error bound will be smaller for Hermite with equal spacing.

**Lagrange basis functions** There are also Lagrange-type basis functions for Hermite interpolation. For example, we can build \( (K + 1)(n + 1) \) polynomials \( H_{jk}(x) \) of degree \( (K + 1)(n + 1) - 1 \), such that

\[ H_{jk}^{(m)}(x_i) = \delta_{ij}\delta_{km}, \quad 0 \leq k \leq K, \quad 0 \leq j \leq n, \]

after which the Hermite interpolant is given by

\[ p(x) = \sum_{k=0}^{K} \sum_{j=0}^{n} f^{(k)}(x_j)H_{jk}(x) \]
and satisfies
\[ p^{(k)}(x_j) = f^{(k)}(x_j). \]

For \( K = 1 \) and \( n = 1 \) we need 4 cubic polynomials satisfying
\[
\begin{align*}
H_{00}(x_0) &= 1, \quad H'_{00}(x_0) = 0, \quad H_{00}(x_1) = 0, \quad H'_{00}(x_1) = 0, \\
H_{10}(x_0) &= 0, \quad H'_{10}(x_0) = 0, \quad H_{10}(x_1) = 1, \quad H'_{10}(x_1) = 0, \\
H_{01}(x_0) &= 0, \quad H'_{01}(x_0) = 1, \quad H_{01}(x_1) = 0, \quad H'_{01}(x_1) = 0, \\
H_{11}(x_0) &= 0, \quad H'_{11}(x_0) = 0, \quad H_{11}(x_1) = 0, \quad H'_{11}(x_1) = 1.
\end{align*}
\]

These conditions are almost satisfied by the squares \( L_j^2(x) \) of the usual Lagrange basis functions, since \( L_j(x_i) = \delta_{ij} \) implies that \( L_j^2(x_i) = \delta_{ij} \) as well. However, these polynomials (a) have degree only \( 2n \) rather than \( 2n + 1 \), and (b) have the wrong derivative values. Hence we should seek \( H_{jk}(x) \) in the form \((a_{jk} + b_{jk}(x - x_j))L_j^2(x)\), after which the constants \( a_{jk} \) and \( b_{jk} \) must satisfy
\[
\begin{align*}
a_{j0} &= 1, \quad b_{j0} = -2L_j'(x_j), \\
a_{j1} &= 0, \quad b_{j1} = 1.
\end{align*}
\]

Thus
\[
H_{j0}(x) = (1 - 2L_j'(x_j))(x - x_j)L_j(x)^2
\]
and
\[
H_{j1}(x) = (x - x_j)L_j(x)^2.
\]

Newton’s formula can also be used to construct these basis functions, because they interpolate known values at the interpolation points.