The Newton interpolation method for the data points 
\((x_0,y_0), (x_1,y_1), \ldots, (x_n,y_n)\)
describes the \(n\)-degree polynomial interpolant as

\[
p(x) = a_0 \cdot \frac{1}{n_0(x)} + a_1 \cdot \frac{(x-x_0)}{n_1(x)} + a_2 \cdot \frac{(x-x_0)(x-x_1)}{n_2(x)} + \cdots + a_n \cdot \frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{n_n(x)}
\]

The coefficients \(\{a_i\}\) are computed using the method of divided differences: For a set of \(x\)-values \(x_i, x_{i+1}, \ldots, x_j\), we define the divided difference as \(f[x_i, \ldots, x_j]\).

The value of this symbol is recursively defined as:

\[
\begin{align*}
\rightarrow f[x_i] &= y_i \quad \text{for } i = 0, \ldots, n \\
\rightarrow f[x_i, x_{i+1}, \ldots, x_j] &= \frac{f[x_i, x_{i+1}, \ldots, x_j] - f[x_i, x_{i+1}, \ldots, x_{j-1}]}{x_j - x_i}
\end{align*}
\]
Once all relevant divided differences have been computed, the coefficients \( c_i \) are given by:

\[
\begin{align*}
  c_0 &= f[x_0] \\
  c_1 &= f[x_0, x_1] \\
  c_2 &= f[x_0, x_1, x_2] \\
  &\vdots \\
  c_n &= f[x_0, x_1, \ldots, x_n]
\end{align*}
\]

Divided differences are usually tabulated as follows:

<table>
<thead>
<tr>
<th>( x_0 )</th>
<th>( f[x_0] )</th>
<th>( f[x_0, x_1] )</th>
<th>( \vdots )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( f[x_1] )</td>
<td>( f[x_0, x_1] )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>( f[x_2] )</td>
<td>( f[x_1, x_2] )</td>
<td>( f[x_0, x_1, x_2] )</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>( f[x_3] )</td>
<td>( f[x_2, x_3] )</td>
<td>( f[x_1, x_2, x_3] )</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>( f[x_4] )</td>
<td>( f[x_3, x_4] )</td>
<td>( f[x_2, x_3, x_4] )</td>
</tr>
</tbody>
</table>

\[\vdots\]
The recursive definition can be implemented directly on the table as follows:

\[ Y = \frac{a-b}{c-d} \]

e.g. \((x_0, y_0) = (-2, -27)\) \((x_1, y_1) = (0, -1)\) \((x_2, y_2) = (1, 0)\)


\[ f(x) = c_0 + c_1 (x-x_0) + c_2 (x-x_0)(x-x_1) + c_3 (x-x_0)(x-x_1)(x-x_2) + c_4 (x-x_0)(x-x_1)(x-x_2)(x-x_3) = \]

\[ = c_0 + (x-x_0) \left[ c_1 + (x-x_1) \left[ c_2 + (x-x_2) \left[ c_3 + (x-x_3) c_4 \right] \right] \right] \]

\[ \phantom{= c_0} \overbrace{\overbrace{\overbrace{\overbrace{Q_4(x)}^{Q_3(x)}}^{Q_2(x)}}^{Q_1(x)}}^{Q_0(x)} \]

\[ P(x) = Q_0(x) \]

\[ \text{recursively:} \quad Q_m(x) = c_m \]

\[ Q_{m-1}(x) = c_{m-1} + (x-x_{m-1}) Q_m(x) \]

The value of \( P(x) = Q_0(x) \) can be evaluated (in linear time) by iterating this recurrence \( n \) times.
We also have:

\[ Q_{n-1}(x) = c_{n-1} + (x - x_{n-1}) Q_n(x) \]

\[ \Rightarrow Q'_{n-1}(x) = Q_n(x) + (x - x_{n-1}) Q'_n(x) \]

Thus, once we have computed all the \( Q_k \)'s we can also compute all the derivatives too. Ultimately, \( P'(x) = Q'_0(x) \).