A cubic spline is a function defined piecewise with each piece being a cubic polynomial. Let the break points (knots) be \( x_1 < x_2 < \ldots < x_n \), and let \( y_1, y_2, \ldots, y_n \) be the data values at these points. We set

\[
\delta_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}.
\]

There are \( n - 1 \) subintervals \( x_k \leq x \leq x_{k+1} \) and a cubic polynomial \( P_k(x) \) defined on each subinterval. Using \( s = x - x_k \), and \( h = x_{k+1} - x_k \), we write the polynomials in the form

\[
P_k(x) = P_k(s) = \frac{3hs^2 - 2s^3}{h^3}y_{k+1} + \frac{h^3 - hs^2 + 2s^3}{h^3}y_k + \frac{s^2(s - h)}{h^2}d_{k+1} + \frac{s(s - h)^2}{h^2}d_k.
\]

With the polynomials written this way, it is easy to verify that \( P_k(x_k) = P_k(s = 0) = y_k \), and \( P_k(x_{k+1}) = P_k(s = h) = y_{k+1} \). Thus \( P = \cup P_k \) also has a continuous derivative. In fact you can verify that \( P_k'(x_k) = P_k'(s = 0) = d_k \), and \( P_k'(x_{k+1}) = P_k'(s = h) = d_{k+1} \).

We still have the \( n \) constants \( d_1, \ldots, d_n \) to determine. If we knew a function \( f(x) \) such that \( f(x_k) = y_k \), we could just set \( d_k = f'(x_k) \). Another approach is to determine the values for the \( d_k \) using the values of the divided differences \( \delta_k \). This is the method used in constructing the Shape-Preserving Piecewise Cubic, described in Moler, p100 and in the MATLAB code `pchip.m`.

In a spline, we impose other conditions on \( P \) at the knots to determine the \( d_k \). We shall require that \( P'' \) be continuous at the interior knots \( x_k, k = 2, \ldots, n-1 \):

\[
P_k''(x_k^+) = P_{k-1}''(x_k^-), \quad k = 2, \ldots, n - 1.
\]

This yields the equations for the \( d_k \) (\( k = 2, \ldots, n - 1 \)):

\[
h_k d_{k-1} + 2(h_k - 1 + h_k)d_k + h_k - 1 d_{k+1} = 3(h_k \delta_{k-1} + h_k \delta_k).
\]

We now have \( n - 2 \) equations for the \( n \) unknowns \( d_k \). To make a system that has a unique solution for the \( d_k \), we must add two equations to the \( n - 2 \) equations (2) or remove two unknowns. This can be done in several ways by imposing extra conditions at the ends.

**Complete Spline** We assign values to \( d_1 \) and \( d_n \) using other outside information. For example, we can interpolate a parabola \( r(x) \) through the data points \( (x_1, y_1), (x_2, y_2), (x_3, y_3) \), and take \( d_1 = r'(x_1) \). Do the same thing at the other end. Since \( d_1 \) and \( d_n \) are considered known, we can put them to the other side of equation in the first equation (\( k = 2 \)) of (2) and in the last equation (\( k = n \)).
n - 1) of (2). This yields the \( n - 2 \times n - 2 \) system \( T d = r \) with \( d = (d_2, \ldots, d_{n-1}) \) and the \( n - 2 \times n - 2 \) matrix \( T \) is

\[
T = \begin{bmatrix}
2(h_2 + h_1) & h_1 & & \\
h_3 & 2(h_3 + h_2) & h_2 & \\
 & & \ddots & \\
h_{n-1} & 2(h_{n-2} + h_{n-1}) & & \\
\end{bmatrix}
\]  

(3)

and the \( n - 2 \times n - 2 \) matrix \( T \) is

and

\[
r = \begin{bmatrix}
3(h_1 \delta_2 + h_2 \delta_1) - d_1 h_2 \\
3(h_2 \delta_3 + h_3 \delta_2) \\
\vdots \\
3(h_{n-2} \delta_{n-1} + h_{n-1} \delta_n) - d_n h_{n-2}
\end{bmatrix}
\]

Natural Spline  We set \( P''_1(x_1) = 0 \) and \( P''_{n-1}(x_n) = 0 \). This yields the two additional equations

\[
2d_1 + d_2 = 3 \delta_1 \quad \text{and} \quad d_{n-1} + 2d_n = 3 \delta_{n-1}.
\]

Combining these two equations with the equations (2), we have the matrix equation \( Sd = r \) where \( d = (d_1, \ldots, d_n) \),

\[
r = 3 \begin{bmatrix}
\delta_1 \\
h_1 \delta_2 + h_2 \delta_1 \\
h_2 \delta_3 + h_3 \delta_2 \\
\vdots \\
h_{n-2} \delta_{n-1} + h_{n-1} \delta_n \\
h_{n-1}
\end{bmatrix}
\]

(4)

and the \( n \times n \) matrix \( S \)

\[
S = \begin{bmatrix}
2 & 1 & & & \\
h_2 & 2(h_2 + h_1) & h_1 & & \\
h_3 & 2(h_3 + h_2) & h_2 & & \\
 & & \ddots & \ddots & \\
h_{n-1} & 2(h_{n-2} + h_{n-1}) & h_{n-2} & & \\
 & & & & 2
\end{bmatrix}
\]

(5)

Not a Knot Spline In this type of spline, we obtain two additional conditions by requiring \( P''' \) to be continuous at \( x_2 \) and at \( x_{n-1} \). This is equivalent to using a single cubic to interpolate the data at \( x_1, x_2 \) and \( x_3 \), and a single cubic to interpolate the data at \( x_{n-2}, x_{n-1} \) and \( x_n \).

If we calculate three derivatives of \( P \) from formula (1), we see that on the \( k^{th} \) subinterval, \( P'''_k \) is the constant

\[
P'''_k(s) = \frac{-12 \delta_k + 6(d_{k+1} + d_k)}{h_k^2}.
\]

(5)
To make \( P''' \) continuous at \( x_1 \), we equate these expressions for \( k = 1 \) and \( k = 2 \). This yields the equation

\[
\h_2^2 (d_1 + d_2 - 2\delta_1) = \h_1^2 (d_2 + d_3 - 2\delta_2),
\]

or

\[
\h_2^2 d_1 + (\h_2^2 - \h_1^2) d_2 - \h_1^2 d_3 = 2\h_2^2 \delta_1 + 2\h_1^2 \delta_2. \tag{6}
\]

From (2) with \( k = 2 \), we have

\[
h_1 d_3 = 3(\delta_1 h_2 + \delta_2 h_1) - h_2 d_1 - 2(h_1 + h_2) d_2.
\]

We substitute this expression in (6) to eliminate \( d_3 \), and we obtain, after dividing by \( h_1 + h_2 \),

\[
h_2 d_1 + (h_1 + h_2) d_2 = r_1 = \frac{(2\h_2^2 + 3h_1 h_2) \delta_1 + 5\h_1^2 \delta_2}{h_1 + h_2}.
\]

We make a similar calculation to make \( P''' \) continuous at \( x_{n-1} \). The resulting \( n \times n \) system of equations for \( d = (d_1, \ldots, d_n) \) is \( Ad = r \) where \( A \) is the \( n \times n \) matrix

\[
A = \begin{bmatrix}
\h_2 & h_1 + h_2 & h_1 \\
\h_2 & 2(h_2 + h_1) & 2(h_3 + h_2) \\
h_3 & 2(h_3 + h_2) & h_2 \\
& & \vdots \\
h_{n-1} & 2(h_{n-2} + h_{n-1}) & h_{n-2} \\
h_n & h_{n-1} + h_{n-2} & h_{n-2}
\end{bmatrix}, \tag{7}
\]

and

\[
r = \begin{bmatrix}
r_1 \\
3(h_1 \delta_2 + h_2 \delta_1) \\
3(h_2 \delta_3 + h_3 \delta_2) \\
\vdots \\
3(h_{n-2} \delta_2 + h_{n-1} \delta_{n-2}) \\
r_n
\end{bmatrix}.
\]

The code `spline.m` of MATLAB uses the not a knot spline.

Note that in three kinds described here, the matrix is tridiagonal, and can the system can be solved very quickly.