Curve fitting: piecewise polynomial interpolation (splines), part 2

The formulas

$$d_k = \frac{c_{k+1} - c_k}{3h_k} \qquad (*)$$

and

$$b_k = g_k - \frac{(2c_k + c_{k+1})}{3} h_k \qquad (\dagger)$$

derived from conditions $\boxed{2}$, $\boxed{3}$ and $\boxed{4}$ for cubic spline interpolation were valid only for $k = 1, \ldots, n-2$. But note that the formulas for d_{n-1} and b_{n-1} that we derived from the boundary condition $S''_{n-1}(x_n) = 0$ and equation $S_{n-1}(x_n) = y_n$ from condition $\boxed{1}$ are actually just $(*)$ and $(\dagger)$ with $k = n-1$, provided we take the undefined coefficient "c_n" to be zero.

This suggests a simpler way of constructing the piecewise cubic interpolant $S(x)$: pretend that there is an nth spline with left endpoint at (x_n, y_n), whose formula is

$$\begin{aligned}
& S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3 \\
\text{so that} \qquad & S'_n(x) = b_n + 2c_n(x - x_n) + 3d_n(x - x_n)^2 \\
& S''_n(x) = 2c_n + 6d_n(x - x_n)
\end{aligned}$$

Therefore $a_n = y_n$, $b_n = S'(x_n)$ and $c_n = \frac{1}{2}S''(x_n)$, i.e. the coefficients of $S_n(x)$ encode the boundary conditions at the right end of the data set.
This fake extra spline has to connect smoothly with $S_{n-1}(x)$ at x_n, so conditions $\boxed{2}$, $\boxed{3}$ and $\boxed{4}$ can be extended to $k = n-1$. Thus formulas $(*)$, $(\dagger)$ and also

$$b_k + (c_k + c_{k+1})h_k = b_{k+1} \qquad (\star)$$

are now valid for $k = 1, \ldots, n-1$. Therefore, upon substituting the expressions for b_k and b_{k+1} from $(\dagger)$ into $(\star)$, we get that

$$h_k c_k + 2(h_k + h_{k+1})c_{k+1} + h_{k+1}c_{k+2} = 3(g_{k+1} - g_k) \quad \text{holds for} \quad k = 1, \ldots, n-1.$$

We write the extended system as $\widetilde{M}\mathbf{c} = \mathbf{rhs}$:

$$\begin{pmatrix}
 & & & & & & & \\
h_1 & 2(h_1+h_2) & h_2 & 0 & \cdots & \cdots & \cdots & 0 \\
0 & h_2 & 2(h_2+h_3) & h_3 & 0 & \cdots & \cdots & 0 \\
0 & 0 & h_3 & 2(h_3+h_4) & h_4 & 0 & \cdots & 0 \\
\vdots & \vdots & & & \ddots & & & \vdots \\
\vdots & \vdots & & & & \ddots & & \vdots \\
 & & & & & & & 0 \\
0 & 0 & 0 & \cdots & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\
 & & & & & & &
\end{pmatrix}
\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ \vdots \\ \vdots \\ c_{n-2} \\ c_{n-1} \\ c_n \end{pmatrix}
=
\begin{pmatrix} \\ 3(g_2 - g_1) \\ 3(g_3 - g_2) \\ \vdots \\ \vdots \\ \vdots \\ 3(g_{n-1} - g_{n-2}) \\ \\ \end{pmatrix}$$

where $\widetilde{M}$ is an $n \times n$ matrix and **rhs** is a length-n column vector.
For "natural" boundary conditions, put $1\ 0\ \cdots\ 0$ as the first row and $0\ \cdots\ 0\ 1$ as the last row of $\widetilde{M}$, and put 0 as both the first and the last entry of **rhs**. This says that $c_1 = 0$ (left boundary condition $S''(x_1) = 0$) and $c_n = 0$ (right boundary condition $S''(x_n) = 0$).

For “clamped” boundary conditions,

$$S'(x_1) = \gamma_{\text{left}} \implies b_1 = \gamma_{\text{left}}$$

and then (†) with $k = 1$ gives

$$b_1 = g_1 - \frac{(2c_1 + c_2)}{3} h_1 = \gamma_{\text{left}}$$

which can be rearranged to

$$2h_1 c_1 + h_1 c_2 = 3(g_1 - \gamma_{\text{left}})$$

Similarly,

$$S'(x_n) = \gamma_{\text{right}} \implies b_n = \gamma_{\text{right}}$$

and (†) with $k = n-1$ gives

$$b_{n-1} = g_{n-1} - \frac{(2c_{n-1} + c_n)}{3} h_{n-1}$$

So ($\star$) with $k = n-1$ becomes

$$\begin{aligned} b_{n-1} + (c_{n-1} + c_n)h_{n-1} &= b_n \\ g_{n-1} - \frac{(2c_{n-1} + c_n)}{3} h_{n-1} + (c_{n-1} + c_n)h_{n-1} &= \gamma_{\text{right}} \\ g_{n-1} + \frac{(c_{n-1} + 2c_n)}{3} h_{n-1} &= \gamma_{\text{right}} \end{aligned}$$

which can be rearranged to

$$h_{n-1} c_{n-1} + 2h_{n-1} c_n = 3(\gamma_{\text{right}} - g_{n-1})$$

Therefore, put $2h_1 \;\; h_1 \;\; 0 \;\cdots\; 0$ as the first row and $0 \;\cdots\; 0 \;\; h_{n-1} \;\; 2h_{n-1}$ as the last row of $\widetilde{M}$, and put $3(g_1 - \gamma_{\text{left}})$ as the first entry and $3(\gamma_{\text{right}} - g_{n-1})$ as the last entry of **rhs**.

When writing code to construct a spline interpolant for a data set of size n, it is simpler to tag on the extra nth spline and then discard the coefficients a_n, b_n etc. at the end.

We will test our code by constructing spline interpolants for the top portion of the following picture of a duck in flight. A sequence of coordinates along the profile that we want to approximate is given in `ducktop.dat`.