# Least squares cubic splines without B-splines

S.K. Lucas

School of Mathematics and Statistics, University of South Australia, Mawson Lakes SA 5095

e-mail: `stephen.lucas@unisa.edu.au`

*Submitted to the Gazette of the Australian Mathematical Society May 2003, Accepted July 2003*

## 1. Justification

Readers of the Gazette will be familiar with the use of least squares to find the polynomial that best fits a set of data points. As with all curve fitting problems, there are many situations where one low order polynomial is inadequate and a high order polynomial fit is inappropriate. A least squares cubic spline would provide a better fit.

There are thousands of articles published on variants of splines, including least squares cubic splines. One of the first least squares articles was de Boor and Rice [1], and a comprehensive explanatory textbook is Dierckx [2]. Unfortunately, every example in the literature and on the web of a least squares cubic spline makes use of B-splines. While B-splines have a certain elegance, they are sufficiently complex to be beyond the typical undergraduate level, and have the disadvantage of being more expensive to evaluate than traditional cubic splines. For example, Schumacker [4] points out that it is more efficient to convert a cubic B-spline to a traditional cubic spline and then evaluate if you require two or more function evaluations per interval. The only exception to using B-splines is Ferguson and Staley [3], where they find a least squares cubic fit to data that enforces continuity of function and first derivative, but not second derivative. Thus, their formulation does not lead to a cubic spline. My colleague, Basil Benjamin, had been using what is essentially the same cubic fit as [3] when fitting smooth curves to train line data. This motivated me to seek the alternative that also enforces continuity of the second derivative.

The aim of this note, then, is to show how least squares cubic splines can be formulated in a straightforward manner using a traditional cubic spline definition. While we shall show that the system of linear equations to be solved is larger than when using B-splines, $(N+1)\times(N+1)$ given $N$ intervals, the theory involved will be much more accessible to readers without advanced numerical analysis experience.

## 2. Derivation

Assume that we are given a set of points $\{(x_i, y_i)\}_{i=1}^{n}$ for which we require a least squares fit. Let $t_1 < t_2 < \cdots < t_{N+1}$ be $N+1$ nodes, where $t_1 \leq x_1$ and $t_{N+1} \geq x_n$. The rest of the t's do not need to be placed at data points. A piecewise cubic function $f(x)$ is defined on the domain $[t_1, t_{N+1}]$ such that $f_i(x) = f(x)$ on $[t_i, t_{i+1}]$ is a polynomial of degree at most three. We wish to find the coefficients of the various cubics such that we minimise the sum of the squares of the errors at the data points, with the constraints that the cubics have continuity of function, first and second derivative, so that $f(x)$ is in fact a cubic spline. Our approach will be the standard one of Lagrange multipliers: to minimise $f(\mathbf{x})$ with constraints $g_i(\mathbf{x}) = 0$, we minimise $f(\mathbf{x}) + \sum \lambda_i g_i(\mathbf{x})$.

Following Ferguson and Staley [3], we define our piecewise cubic as

$$f(x) = (2u_i(x)+1)v_i^2(x)z_i + u_i^2(x)(1-2v_i(x))z_{i+1} + h_i u_i(x)v_i^2(x)z_i' + h_i u_i^2 v_i(x)z_{i+1}', \qquad (1)$$

for $x \in [t_i, t_{i+1}]$, $i = 1, 2, \ldots, N$, where

$$h_i = t_{i+1} - t_i, \quad u_i(x) = (x - x_i)/h_i, \quad \text{and } v_i = u_i(x) - 1. \tag{2}$$

This ensures that $f(t_i) = z_i$ and $f'(t_i) = z_i'$ for $i = 2, 3, \ldots, N-1$. In other words, the function and its first derivative are already continuous. While not a standard way of writing a cubic, this form will make the following analysis far more straightforward. If you prefer a standard cubic for efficient nested evaluation, (1) can be rewritten on $[t_i, t_{i+1}]$ as

$$\begin{aligned} f(x) \;=\; & z_i + z_i'(x - t_i) + \left[-\frac{3}{h_i^2}(z_i - z_{i+1}) - \frac{1}{h_i}(2z_i' + z_{i+1}')\right](x - t_i)^2 + \\ & \left[\frac{2}{h_i^3}(z_i - z_{i+1}) + \frac{1}{h_i^2}(z_i' + z_{i+1}')\right](x - t_i)^3. \end{aligned} \tag{3}$$

The constraints that ensure $f(x)$ is a cubic spline are continuity of the second derivative at $t_i$, $i = 2, 3, \ldots, N$. This reduces to

$$h_i z_{i-1}' + 2(h_{i-1} + h_i)z_i' + h_{i-1}z_{i+1}' - 3\frac{h_{i-1}}{h_i}(z_{i+1} - z_i) - 3\frac{h_i}{h_{i-1}}(z_i - z_{i-1}) = 0 \tag{4}$$

for $i = 2, 3, \ldots, N$. Now let us assume that the data $\{(x_i, y_i)\}_{i=1}^n$ is ordered on $x$ so that we can easily identify how many points $(n_i)$ are in each interval and where in the list $(p_i)$ they begin. Under these conditions, the constrained minimisation problem is reduced to minimising

$$\begin{aligned} S \;=\; & \sum_{i=1}^{N}\sum_{j=p_i}^{p_i+n_i-1}(y_j - \alpha_{ij}z_i - \beta_{ij}z_{i+1} - \gamma_{ij}z_i' - \delta_{ij}z_{i+1}')^2 + \\ & \sum_{i=2}^{N}\lambda_i\left[h_i z_{i-1}' + 2(h_{i-1} + h_i)z_i' + h_{i-1}z_{i+1}' - 3\frac{h_{i-1}}{h_i}(z_{i+1} - z_i) - 3\frac{h_i}{h_{i-1}}(z_i - z_{i-1})\right], \end{aligned} \tag{5}$$

where

$$\begin{aligned} \alpha_{ij} &= (2u_i(x_j) + 1)v_i^2(x_j), & \beta_{ij} &= u_i^2(x_j)(1 - 2v_i(x_j)), \\ \gamma_{ij} &= h_i u_i(x_j)v_i^2(x_j), & \delta_{ij} &= h_i u_i^2(x_j)v_i(x_j). \end{aligned} \tag{6}$$

The $\{\lambda_i\}_{i=2}^N$ are the Lagrange multipliers, and combined with the unknowns $\{z_i, z_i'\}_{i=1}^{N+1}$ there are $3N + 1$ unknowns. As with any least squares problem, we can take the partial derivatives of $S$ with respect to each unknown, being careful to recognise that each unknown occurs in several terms on the right hand side of (5), and equate to zero. This leads to the following linear system of equations:

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}^T & \mathcal{C}^T \\ \mathcal{B} & \mathcal{E} & \mathcal{F}^T \\ \mathcal{C} & \mathcal{F} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{z} \\ \mathbf{z}' \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathcal{D} \\ \mathcal{G} \\ \mathbf{0} \end{pmatrix}, \tag{7}$$

where (with the abbreviation $\sum^k$ for $\sum_{j=p_k}^{p_k+n_k-1}$)

- $\mathcal{A}$ is the $(N+1) \times (N+1)$ symmetric tridiagonal matrix with main diagonal $2\sum^1 \alpha_{1j}^2$, $2(\sum^2 \alpha_{2j}^2 + \sum^1 \beta_{1j}^2)$, $2(\sum^3 \alpha_{3j}^2 + \sum^2 \beta_{2j}^2)$, …, $2(\sum^N \alpha_{Nj}^2 + \sum^{N-1} \beta_{N-1,j}^2)$, $2\sum^N \beta_{Nj}^2$, and codiagonal $2\sum^1 \alpha_{1j}\beta_{1j}$, $2\sum^2 \alpha_{2j}\beta_{2j}$, …, $2\sum^N \alpha_{Nj}\beta_{Nj}$,
- $\mathcal{B}$ is the $(N+1)\times(N+1)$ tridiagonal matrix with main diagonal $2\sum^1 \alpha_{1j}\gamma_{1j}$, $2(\sum^2 \alpha_{2j}\gamma_{2j} + \sum^1 \beta_{1j}\delta_{1j})$, $2(\sum^3 \alpha_{3j}\gamma_{3j} + \sum^2 \beta_{2j}\delta_{2j})$, …, $2(\sum^N \alpha_{Nj}\gamma_{Nj} + \sum^{N-1} \beta_{N-1,j}\delta_{N-1,j})$, $2\sum^N \beta_{Nj}\delta_{Nj}$, upper diagonal $2\sum^1 \beta_{1j}\gamma_{1j}$, $2\sum^2 \beta_{2j}\gamma_{2j}$, …, $2\sum^N \beta_{Nj}\gamma_{Nj}$, and lower diagonal $2\sum^1 \alpha_{1j}\delta_{1j}$, $2\sum^2 \alpha_{2j}\delta_{2j}$, …, $2\sum^N \alpha_{Nj}\delta_{Nj}$,

- $\mathcal{C}$ is the $(N-1)\times(N+1)$ upper diagonal matrix with $c_{ii} = 3h_{i+1}/h_i$, $c_{i,i+1} = 3(h_i/h_{i+1} - h_{i+1}/h_i)$, and $c_{i,i+2} = -3h_i/h_{i+1}$ for $i = 1, 2, \ldots, N-1$, all other terms zero,
- $\mathcal{D}$ is the $(N+1)\times 1$ vector with terms $2\sum^1 y_j\alpha_{1j}$, $2(\sum^2 y_j\alpha_{2j} + \sum^1 y_j\beta_{1j})$, $2(\sum^3 y_j\alpha_{3j} + \sum^2 y_j\beta_{2j})$, …, $2(\sum^N y_j\alpha_{Nj} + \sum^{N-1} y_j\beta_{N-1,j})$, $2\sum^N y_j\beta_{Nj}$,
- $\mathcal{E}$ is the $(N+1)\times(N+1)$ symmetric tridiagonal matrix with main diagonal $2\sum^1 \gamma_{1j}^2$, $2(\sum^2 \gamma_{2j}^2 + \sum^1 \delta_{1j}^2)$, $2(\sum^3 \gamma_{3j}^2 + \sum^2 \delta_{2j}^2)$, …, $2(\sum^N \gamma_{Nj}^2 + \sum^{N-1} \delta_{N-1,j}^2)$, $2\sum^N \delta_{Nj}^2$, and codiagonal $2\sum^1 \gamma_{1j}\delta_{1j}$, $2\sum^2 \gamma_{2j}\delta_{2j}$, …, $2\sum^N \gamma_{Nj}\delta_{Nj}$,
- $\mathcal{F}$ is the $(N-1)\times(N+1)$ upper diagonal matrix with $f_{ii} = h_{i+1}$, $f_{i,i+1} = 2(h_i + h_{i+1})$, and $f_{i,i+2} = h_i$ for $i = 1, 2, \ldots, N-1$, all other terms zero,
- $\mathcal{G}$ is the $(N+1)\times 1$ vector $2\sum^1 y_j\gamma_{ij}$, $2(\sum^2 y_j\gamma_{2j} + \sum^1 y_j\delta_{1j})$, $2(\sum^3 y_j\gamma_{3j} + \sum^2 y_j\delta_{2j})$, …, $2(\sum^N y_j\gamma_{Nj} + \sum^{N-1} y_j\delta_{N-1,j})$, $2\sum^N y_j\delta_{nj}$,
- $\mathbf{z} = [z_1, z_2, \ldots, z_{N+1}]^T$, $\mathbf{z}' = [z_1', z_2', \ldots, z_{N+1}']^T$, $\boldsymbol{\lambda} = [\lambda_2, \lambda_3, \ldots, \lambda_N]^T$, and
- the zero matrix on the left hand side is $(N-1)\times(N-1)$, and the zero vector on the right hand side is $(N-1)\times 1$.

## 3. Implementation

Setting up and solving (7) by Gaussian elimination is not difficult, particularly if appropriate intermediate variables are used. Equation (3) can then be used to output the solution as standard piecewise cubics. For $N$ piecewise cubics we require at least $3N+1$ data points, but there is no constraint on the position of these points. In fact it is perfectly acceptable (if unusual) to have intervals with no data points whatsoever.

Pseudocode for a function that implements this formulation of a least squares cubic spline is listed in figure 1. Figure 2 shows two examples of least squares cubic spline fits to data using this algorithm. The first is the classic titanium heat data used to test curve fitting routines, where spline intervals have been chosen for a good fit. The appropriate positioning of interval endpoints continues to be an area of active research. The second is a simple fit to sinusoidal data with small Gaussian noise, where data is not available in each interval. As one would expect, the solutions are exactly those obtained using a B-spline approach.

## References

**[1]** C. de Boor and J.R. Rice, Least squares cubic spline approximation I – fixed knots, *Technical Report CSD-TR 20*, Computer Sciences, Purdue University (1968). Also available at `ftp://ftp.cs.wisc.edu/Approx/tr20.pdf`

**[2]** P. Dierckx, *Curve and surface fitting with splines*, Oxford University Press, 1993.

**[3]** J. Ferguson and P.A. Staley, Least squares piecewise cubic curve fitting, *Comm. ACM* **16** 380–382 (1973).

**[4]** L.L. Schumaker, *Spline functions: basic theory*, John Wiley and Sons, New York, 1981.

Algorithm to find a least squares cubic spline fit to data $(x_i, y_i)_{i=1}^n$ on intervals $[t_i, t_{i+1}]_{i=1}^N$.

Input: Data $(x_i, y_i)_{i=1}^n$ and interval endpoints $\{t_i\}_{i=1}^{N+1}$, number of data points $n$ and intervals $N$.
Output: Function $\{z_i\}_{i=1}^{N+1}$ and derivative $\{z'_i\}_{i=1}^{N+1}$ data at interval endpoints.

```
procedure least_squares_spline($x, y, n, t, N, z, z'$)
    sort($x, y$)                // sort data points on x coordinates
    for $i = 1, N$
        $h_i = t_{i+1} - t_i$
    endfor
    $p_1 = 1$, $pn_1 = 0$, $j = 1$
    for $i = 1, n$              // identify which data is in which interval
        if $x_i \le t_{j+1}$
            $pn_j = pn_j + 1$
        else
            $j = j + 1$, $p_j = i$, $pn_j = 0$
            while $x_i > t_{j+1}$           // deal with intervals with no points
                $j = j + 1$, $p_j = i$, $pn_j = 0$
            endwhile
            $pn_j = 1$
        endif
    endfor
    Set $taa$, $tab$, $tag$, $tad$, $tbb$, $tbg$, $tbd$, $tgg$, $tgd$, $tdd$ to zero vectors of length $N$
    Set $\mathcal{D}$ and $\mathcal{G}$ to zero vectors of length $N + 1$
    for $i = 1, N$              // Set up intermediate sums and vectors
        for $j = p_i, p_i + pn_i - 1$
            $u = (x_j - t_i)/h_i$, $v = u - 1$, $\alpha = (2u + 1)v^2$, $\beta = u^2(1 - 2v)$, $\gamma = h_i uv^2$, $\delta = h_i u^2 v$
            $taa_i = taa_i + \alpha^2$, $tab_i = tab_i + \alpha\beta$, $tag_i = tag_i + \alpha\gamma$, $tad_i = tad_i + \alpha\delta$, $tbb_i = tbb_i + \beta^2$
            $tbg_i = tbg_i + \beta\gamma$, $tbd_i = tbd_i + \beta\delta$, $tgg_i = tgg_i + \gamma^2$, $tgd_i = tgd_i + \gamma\delta$, $tdd_i = tdd_i + \delta^2$
            $\mathcal{D}_i = \mathcal{D}_i + 2y_j\alpha$, $\mathcal{D}_{i+1} = \mathcal{D}_{i+1} + 2y_j\beta$, $\mathcal{G}_i = \mathcal{G}_i + 2y_j\gamma$, $\mathcal{G}_{i+1} = \mathcal{G}_{i+1} + 2y_j\delta$
        endfor
    endfor
    Set $\mathcal{A}$, $\mathcal{B}$, $\mathcal{E}$ to zero matrices of size $(N + 1) \times (N + 1)$
    Set $\mathcal{C}$, $\mathcal{F}$ to zero matrices of size $(N - 1) \times (N + 1)$
    for $i = 1, N$              // Set up the intermediate matrices
        $\mathcal{A}_{ii} = \mathcal{A}_{ii} + 2taa_i$, $\mathcal{A}_{i+1,i+1} = 2tbb_i$, $\mathcal{A}_{i,i+1} = 2tab_i$, $\mathcal{A}_{i+1,i} = \mathcal{A}_{i,i+1}$,
        $\mathcal{B}_{ii} = \mathcal{B}_{ii} + 2tag_i$, $\mathcal{B}_{i+1,i+1} = 2tbd_i$, $\mathcal{B}_{i,i+1} = 2tbg_i$, $\mathcal{B}_{i+1,i} = 2tad_i$
        $\mathcal{E}_{ii} = \mathcal{E}_{ii} + 2tgg_i$, $\mathcal{E}_{i+1,i+1} = 2tdd_i$, $\mathcal{E}_{i,i+1} = 2tgd_i$, $\mathcal{E}_{i+1,i} = \mathcal{E}_{i,i+1}$
    endfor
    for $i = 1, N - 1$
        $\mathcal{C}_{ii} = 3h_{i+1}/h_i$, $\mathcal{C}_{i,i+2} = -3h_i/h_{i+1}$, $\mathcal{C}_{i,i+1} = -\mathcal{C}_{ii} - \mathcal{C}_{i,i+2}$
        $\mathcal{F}_{ii} = h_{i+1}$, $\mathcal{F}_{i,i+1} = 2(h_i + h_{i+1})$, $\mathcal{F}_{i,i+2} = h_i$
    endfor
    Set up and solve $\begin{pmatrix} \mathcal{A} & \mathcal{B}^T & \mathcal{C}^T \\ \mathcal{B}^T & \mathcal{E} & \mathcal{F}^T \\ \mathcal{C} & \mathcal{F} & \mathbf{0} \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathcal{D} \\ \mathcal{G} \\ \mathbf{0} \end{pmatrix}$
    Extract $z = \{x_i\}_{i=1}^{N+1}$, $z' = \{x_i\}_{i=N+2}^{2N+2}$
end(least_squares_spline)
```

Figure 1: Pseudocode for the least squares cubic spline algorithm.

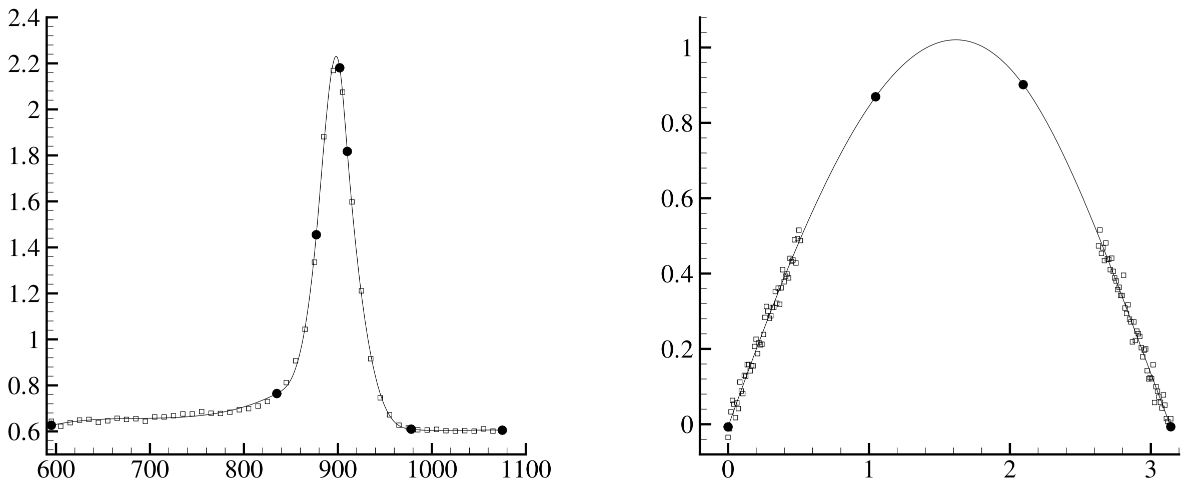

Figure 2: Examples of fitting least squares cubic splines to data, open squares are data points, closed circles are spline interval endpoints.