SMOOTHING WITH CUBIC SPLINES

by

D.S.G. Pollock

Queen Mary and Westfield College,
The University of London

A spline function is a curve constructed from polynomial segments that are subject to conditions or continuity at their joints. In this paper, we shall present the algorithm of the cubic smoothing spline and we shall justify its use in estimating trends in time series.

Considerable effort has been devoted over several decades to developing the mathematics of spline functions. Much of the interest is due to the importance of splines in industrial design. In statistics, smoothing splines have been used in fitting curves to data ever since workable algorithms first became available in the late sixties—see Schoenberg [9] and Reinsch [8]. However, many statisticians have felt concern at the apparently arbitrary nature in this device.

The difficulty is in finding an objective criterion for choosing the value of the parameter which governs the trade-off between smoothness of the curve and its closeness to the data points. At one extreme, where the smoothness is all that matters, the spline degenerates to the straight line of an ordinary linear least-squares regression. At the other extreme, it becomes the interpolating spline which passes through each of the data points. It appears to be a matter of judgment where in the spectrum between these two extremes the most appropriate curve should lie.

One attempt at overcoming this arbitrariness has led to the criterion of cross-validation which was first proposed by Craven and Wahba [6]. Here the underlying notion is that the degree of smoothness should be chosen so as to make the spline the best possible predictor of any points in the data set to which it has not been fitted. Instead of reserving a collection of data points for the sole purpose of making this choice, it has been proposed that each of the available points should be left out in turn while the spline is fitted to the remainder. For each omitted point, there is then a corresponding error of prediction; and the optimal degree of smoothing is that which results in the minimum sum of squares of the prediction errors.

To find the optimal degree of smoothing by the criterion of cross-validation can require an enormous amount of computing. An alternative procedure, which has emerged more recently, is based on the notion that the spline with an appropriate smoothing parameter represents the optimal predictor of the

path of a certain stochastic differential equation of which the observations are affected by noise. This is a startling result, and it provides a strong justification for the practice of representing trends with splines. The optimal degree of smoothing now becomes a function of the parameters of the underling stochastic differential equation and of the parameters of the noise process; and therefore the element of judgment in fitting the curve is eliminated.

We shall begin this account by establishing the algorithm for an ordinary interpolating spline. Thereafter, we shall give a detailed exposition of the classical smoothing spline of which the degree of smoothness is a matter of choice. In the final section, we shall give an account of a model-based method of determining an optimal degree of smoothing.

It should be emphasised that a model-based procedure for determining the degree of smoothing will prove superior to a judgmental procedure only if the model has been appropriately specified. The specification of a model is itself a matter of judgment.

Cubic Spline Interpolation

Imagine that we are given a set of co-ordinates $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ of the function $y = y(x)$ where the values of x are in ascending order. Our object is to bridge the gap between adjacent points (x_i, y_i), (x_{i+1}, y_{i+1}) using the cubic functions $S_i; i = 0, \ldots, n-1$ so as to piece together a curve with continuous first and second derivatives. Such a curve, which is described as a cubic spline, is the mathematical equivalent of a draughtsman's spline which is a thin strip of flexible wood used for drawing curves in engineering work. The junctions of the cubic segments, which correspond to the points at which the draughtsman's spline would be fixed, are known as knots or nodes.

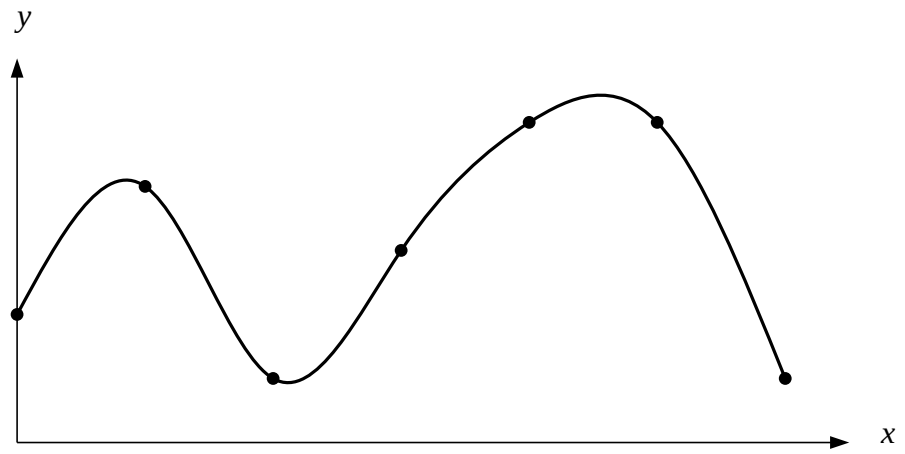

Figure 1. A cubic spline.

We can express the function S_i as

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \tag{1}$$

where x ranges from x_i to x_{i+1}.

The first and second derivatives of this function are

$$\begin{aligned} S_i'(x) &= 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \quad \text{and} \\ S_i''(x) &= 6a_i(x - x_i) + 2b_i. \end{aligned} \tag{2}$$

The condition that the adjacent functions S_{i-1} and S_i for $i = 1, \ldots, n$ should meet at the point (x_i, y_i) is expressed in the equation

$$\begin{aligned} &S_{i-1}(x_i) = S_i(x_i) = y_i \quad \text{or, equivalently,} \\ &a_{i-1}h_{i-1}^3 + b_{i-1}h_{i-1}^2 + c_{i-1}h_{i-1} + d_{i-1} = d_i = y_i, \end{aligned} \tag{3}$$

where $h_{i-1} = x_i - x_{i-1}$. The condition that the first derivatives should be equal at the junction is expressed in the equation

$$\begin{aligned} &S_{i-1}'(x_i) = S_i'(x_i) \quad \text{or, equivalently,} \\ &3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1} = c_i; \end{aligned} \tag{4}$$

and the condition that the second derivatives should be equal is expressed as

$$\begin{aligned} &S_{i-1}''(x_i) = S_i''(x_i) \quad \text{or, equivalently,} \\ &6a_{i-1}h_{i-1} + 2b_{i-1} = 2b_i. \end{aligned} \tag{5}$$

We also need to specify the conditions which prevail at the end points (x_0, y_0) and (x_n, y_n). We can set the first derivatives of the cubic functions at these points to the values of the corresponding derivatives of $y = y(x)$ thus:

$$\begin{aligned} S_0'(x_0) &= c_0 \\ &= y'(x_0) \end{aligned} \quad \text{and} \quad \begin{aligned} S_{n-1}'(x_n) &= c_n \\ &= y'(x_n). \end{aligned} \tag{6}$$

This is described as clamping the spline. By clamping the spline, we are introducing additional information about the function $y = y(x)$; and, therefore, we can expect a better approximation. However, extra information of an equivalent nature can often be obtained by assessing the function at additional points close to the ends.

If we leave the ends free, then the conditions

$$\begin{aligned} S_0''(x_0) &= 2b_0 \\ &= 0 \end{aligned} \quad \text{and} \quad \begin{aligned} S_{n-1}''(x_n) &= 2b_n \\ &= 0 \end{aligned} \tag{7}$$

will prevail. These imply that the spline is linear when it passes through the end points. We are likely to use the latter conditions when the information about the first derivatives of the function $y = y(x)$ is hard to come by.

We shall begin by treating the case of the natural spline which has free ends. In this case, we know the values of b_0 and b_n, and we can begin by determining the remaining second-degree parameters $b_1, \ldots, b_{n-1}$ from the data points $y_0, \ldots, y_n$ and from the conditions of continuity. Once we have found the values for the second-degree parameters, we can proceed to determine the values of the remaining parameters of the cubic segments.

Consider therefore the following four conditions relating to the ith segment:

$$\begin{aligned} &\text{(i)} \quad S_i(x_i) = y_i, && \text{(ii)} \quad S_i(x_{i+1}) = y_{i+1}, \\ &\text{(iii)} \quad S_i''(x_i) = 2b_i, && \text{(iv)} \quad S_i''(x_{i+1}) = 2b_{i+1}. \end{aligned} \tag{8}$$

If b_i and b_{i+1} were known in advance, as they would be in the case of $n = 1$, then these conditions would serve to specify uniquely the four parameters of S_i. In the case of $n > 1$, the conditions of first-order continuity provide the necessary link between the segments which enables us to determine the parameters $b_1, \ldots, b_{n-1}$ simultaneously.

The first of the four conditions specifies that

$$d_i = y_i. \tag{9}$$

The second condition specifies that $a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i = y_{i+1}$, whence we get

$$c_i = \frac{y_{i+1} - y_i}{h_i} - a_i h_i^2 - b_i h_i. \tag{10}$$

The third condition may be regarded as an identity. The fourth condition specifies that $6a_i h_i + 2b_i = 2b_{i+1}$, which gives

$$a_i = \frac{b_{i+1} - b_i}{3h_i}. \tag{11}$$

Putting this into (10) gives

$$c_i = \frac{(y_{i+1} - y_i)}{h_i} - \frac{1}{3}(b_{i+1} + 2b_i)h_i; \tag{12}$$

and thus we have succeeded in expressing the parameters of the ith segment in terms of the second-order parameters b_{i+1}, b_i and the data values y_{i+1}, y_i.

The condition $S'_{i-1}(x_i) = S'_i(x_i)$ of first-order continuity, which is to be found under (4), can now be rewritten with the help of equations (11) and (12) to give

$$b_{i-1}h_{i-1} + 2b_i(h_{i-1} + h_i) + b_{i+1}h_i = \frac{3}{h_i}(y_{i+1} - y_i) - \frac{3}{h_{i-1}}(y_i - y_{i-1}), \tag{13}$$

By letting i run from 1 to $n-1$ in this equation and by taking account of the end conditions $b_0 = b_n = 0$, we obtain a tridiagonal system of $n-1$ equations in the form of

$$\begin{bmatrix} p_1 & h_1 & 0 & \dots & 0 & 0 \\ h_1 & p_2 & h_2 & \dots & 0 & 0 \\ 0 & h_2 & p_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & p_{n-2} & h_{n-2} \\ 0 & 0 & 0 & \dots & h_{n-2} & p_{n-1} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \vdots \\ q_{n-2} \\ q_{n-1} \end{bmatrix}, \tag{14}$$

where

$$\begin{aligned} p_i &= 2(h_{i-1} + h_i) = 2(x_{i+1} - x_{i-1}) \quad \text{and} \\ q_i &= \frac{3}{h_i}(y_{i+1} - y_i) - \frac{3}{h_{i-1}}(y_i - y_{i-1}). \end{aligned} \tag{15}$$

These can be reduced by Gaussian elimination to a bidiagonal system

$$\begin{bmatrix} p'_1 & h_1 & 0 & \dots & 0 & 0 \\ 0 & p'_2 & h_2 & \dots & 0 & 0 \\ 0 & 0 & p'_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & p'_{n-2} & h_{n-2} \\ 0 & 0 & 0 & \dots & 0 & p'_{n-1} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} q'_1 \\ q'_2 \\ q'_3 \\ \vdots \\ q'_{n-2} \\ q'_{n-1} \end{bmatrix}, \tag{16}$$

which may be solved by backsubstitution to obtain the values $b_1, \dots, b_{n-1}$. The values of $a_i : i = 0, \dots, n-1$ can be obtained from (11). The value of c_0 can be obtained from (12) and then the remaining values $c_i; i = 1, \dots, n-1$ can be generated by a recursion based on the equation

$$c_i = (b_i + b_{i-1})h_{i-1} + c_{i-1}, \tag{17}$$

which comes from substituting into equation (4) the expression for a_{i-1} given by (11).

The following procedure calculates the parameters of a cubic spline of which the ends have been left free in accordance with the conditions under (7):

(18)
```
procedure CubicSplines(var S : SplineVec;
                       n : integer);
  var
    i : subs;
    h, p, q, b : vectors;

  begin {CubicSplines}

    h[0] := S[1].x − S[0].x;
    for i := 1 to n − 1 do
      begin
        h[i] := S[i + 1].x − S[i].x;
        p[i] := 2 ∗ (S[i + 1].x − S[i − 1].x);
        q[i] := 3 ∗ (S[i + 1].y − S[i].y)/h[i] − 3 ∗ (S[i].y − S[i − 1].y)/h[i − 1]
      end;

  {Gaussian Elimination}
    for i := 2 to n − 1 do
      begin
        p[i] := p[i] − h[i − 1] ∗ h[i − 1]/p[i − 1];
        q[i] := q[i] − q[i − 1] ∗ h[i − 1]/p[i − 1]
      end;

  {Backsubstitution}
    b[n − 1] := q[n − 1]/p[n − 1];
    for i := 2 to n − 1 do
      b[n − i] := (q[n − i] − h[n − i] ∗ b[n − i + 1])/p[n − i];

  {Spline Parameters}
    S[0].a := b[1]/(3 ∗ h[0]);
    S[0].b := 0;
    S[0].c := (S[1].y − S[0].y)/h[0] − b[1] ∗ h[0]/3;
    S[0].d := y[0];
    S[n].b := 0;

    for i := 1 to n − 1 do
      begin
        S[i].a := (b[i + 1] − b[i])/(3 ∗ h[i]);
        S[i].b := b[i];
        S[i].c := (b[i] + b[i − 1]) ∗ h[i − 1] + S[i − 1].c;
```

```
        S[i].d := y[i];
      end;

    end; {CubicSplines}
```

The procedure must be placed in an environment containing the following type statements:

(19)
```
type
  SplineParameters = record
      a, b, c, d, x, y : real
    end;
  SplineVec = array[0..dim] of SplineParameters;
```

At the beginning of the procedure, the record $S[i]$ contains only the values of x_i and y_i which are held as $S[i].x$ and $S[i].y$ respectively. At the conclusion of the procedure, the parameters a_i, b_i, c_i, d_i of the ith cubic segment are held in $S[i].a$, $S[i].b$, $S[i].c$ and $S[i].d$ respectively.

Now let us consider the case where the ends of the spline are clamped. Then we know the values of c_0 and c_n, and we may begin by determining the remaining first-degree parameters $c_1, \ldots, c_{n-1}$ from the data points $y_0, \ldots, y_n$ and from the continuity conditions. Consider therefore the following four conditions relating to the ith segment:

(20)
$$\begin{aligned} &\text{(i)} \quad S_i(x_i) = y_i, && \text{(ii)} \quad S_i(x_{i+1}) = y_{i+1}, \\ &\text{(iii)} \quad S_i'(x_i) = c_i, && \text{(iv)} \quad S_i'(x_{i+1}) = c_{i+1}. \end{aligned}$$

If c_i and c_{i+1} were known in advance, as they would be in the case of $n = 1$, then these four conditions would serve to specify the parameters of the segment. The first and second conditions, which are the same as in the case of the natural spline, lead to equation (10). The third condition is an identity, whilst the fourth condition specifies that

$$c_{i+1} = 3a_i h_i^2 + 2b_i h_i + c_i. \tag{21}$$

The equations (10) and (21) can be solved simultaneously to give

$$a_i = \frac{1}{h_i^2}(c_i + c_{i+1}) + \frac{2}{h_i^3}(y_i - y_{i+1}) \tag{22}$$

and

$$b_i = \frac{3}{h_i^2}(y_{i+1} - y_i) - \frac{1}{h_i}(c_{i+1} + 2c_i). \tag{23}$$

The condition $S''_{i-1}(x_i) = S''_i(x_i)$ of second-order continuity, which is to be found under (5), can now be rewritten with the help of equations (22) and (23) to give

$$\frac{c_{i-1}}{h_{i-1}} + 2\Big(\frac{1}{h_{i-1}} + \frac{1}{h_i}\Big)c_i + \frac{c_{i+1}}{h_i} = \frac{3}{h_{i-1}^2}(y_i - y_{i-1}) + \frac{3}{h_i^2}(y_{i+1} - y_i). \tag{24}$$

This is similar to the expression under (13); and, by letting i run from 1 to $n-1$, we can obtain the following system of equations:

$$\begin{bmatrix} f_1 & h_1^{-1} & 0 & \dots & 0 & 0 \\ h_1^{-1} & f_2 & h_2^{-1} & \dots & 0 & 0 \\ 0 & h_2^{-1} & f_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & f_{n-2} & h_{n-2}^{-1} \\ 0 & 0 & 0 & \dots & h_{n-2}^{-1} & f_{n-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} g_1 - c_0 h_0^{-1} \\ g_2 \\ g_3 \\ \vdots \\ g_{n-2} \\ g_{n-1} - c_n h_{n-1}^{-1} \end{bmatrix}, \tag{25}$$

where

$$\begin{aligned} f_i &= 2(h_{i-1}^{-1} - h_i^{-1}) \quad \text{and} \\ g_i &= \frac{3}{h_{i-1}^2}(y_i - y_{i-1}) + \frac{3}{h_i^2}(y_{i+1} - y_i). \end{aligned} \tag{26}$$

These may be solved for the values $c_1, \dots, c_{n-1}$ in the same manner as the equations under (14) are solved for $b_1, \dots, b_{n-1}$, by reducing the tridiagonal matrix to a bidiagonal matrix and then using a process of backsubstitution. The values $a_0, \dots, a_{n-1}$ may then be obtained from equation (22). The value b_0 can be obtained from (23), and then the remaining values $b_1, \dots, b_n$ can be generated using a recursion based on the equation

$$b_i = b_{i-1} + 3a_{i-1}h_{i-1}, \tag{27}$$

which comes from (5).

The alternative procedure for calculating the clamped spline requires us to extend the system of (14) so as to accomodate the additional information concerning the values of the first-derivatives at the endpoints. Since the conditions under (7) no longer prevail, there are two more parameters to be determined.

The value of the derivative at x_0 affects the parameters of the spline via the equation

$$y'_0 = c_0 = \frac{y_1 - y_0}{h_0} - \frac{1}{3}(b_1 + 2b_0)h_0, \tag{28}$$

which comes from combining the first condition under (6) with the equation under (12). This becomes

$$p_0 b_0 + h_0 b_1 = q_0 \tag{29}$$

when we define

$$p_0 = 2h_0 \quad \text{and} \quad q_0 = \frac{3}{h_0}(y_1 - y_0) - 3y'_0. \tag{30}$$

The value of the derivative at x_n affects the parameters of the spline via the equation

$$y'_n = c_n = 3a_{n-1}h_{n-1}^2 + 2b_{n-1}h_{n-1} + c_{n-1}, \tag{31}$$

which comes from combining the second contition under (6) with the condition under (4). Using (11) and (12), we can rewrite this as

$$y'_n - \frac{(y_n - y_{n-1})}{h_{n-1}} = \frac{2}{3}b_n h_{n-1} + \frac{1}{3}b_{n-1}h_{n-1} \tag{32}$$

which becomes

$$h_{n-1}b_{n-1} + p_n b_n = q_n \tag{33}$$

when we define

$$p_n = 2h_n \quad \text{and} \quad q_n = 3y'_n - \frac{3}{h_{n-1}}(y_n - y_{n-1}). \tag{34}$$

The extended system can now be written as

$$\begin{bmatrix} p_0 & h_0 & 0 & \dots & 0 & 0 \\ h_0 & p_1 & h_1 & \dots & 0 & 0 \\ 0 & h_1 & p_2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & p_{n-1} & h_{n-1} \\ 0 & 0 & 0 & \dots & h_{n-1} & p_n \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ \vdots \\ q_{n-1} \\ q_n \end{bmatrix}. \tag{35}$$

Cubic Splines and Bézier Curves

Parametric cubic splines have been much used in the past in ship-building and aircraft design and they have been used, to a lesser extent, in the design of car bodies. However, their suitability to an iterative design process is limited

by the fact that, if the location of one of the knots is altered, then the whole spline must be recalculated. In recent years, cubic splines have been replaced increasingly in computer-aided design applications by the so-called cubic Bézier curve.

A testimony to the versatility of cubic Bézier curves is the fact that the PostScript [2] page-description language, which has been used in constructing the letter forms on these pages, employs Bézier curve segments exclusively in constructing curved paths, including very close approximations to circles.

The usual parametrisation of a Bézier curve differs from the parametrisation of the cubic polynomial to be found under (1). Therefore, in order to make use of the Bézier function provided by a PostScript-compatible printer, we need to establish the correspondence between the two sets of parameters. The Bézier function greatly facilitates the plotting of functions which can be represented exactly or approximately by cubic segments.

The curve-drawing method of Bézier [4], [5] is based on a classical method of approximation known as the Bernstein polynomial approximation. Let $f(t)$ with $t \in [0, 1]$ be an arbitrary real-valued function taking the values $f_k = f(t_k)$ at the points $t_k = k/n; k = 0, \ldots, n$ which are equally spaced in the interval $[0, 1]$. Then the Bernstein polynomial of degree n is defined by

$$B_n(t) = \sum_{k=0}^{n} f_k \frac{n!}{k!(n-k)!} t^k (1-t)^{n-k}. \tag{36}$$

The coefficients in this sum are just the terms of the expansion of the binomial

$$\big\{t + (1-t)\big\}^n = \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} t^k (1-t)^{n-k}; \tag{37}$$

from which it can be seen that the sum of the coefficients is unity.

Bernstein [3] used this polynomial in a classic proof of the Weierstrass approximation theorem [12] which asserts that, for any $\epsilon > 0$, there exists a polynomial $P_n(t)$ of degree $n = n(\epsilon)$ such that $|f(t) - P_n(t)| < \epsilon$. The consequence of Bernstein's proof is that $B_n(t)$ converges uniformly to $f(t)$ in $[0, 1]$ as $n \to \infty$.

The restriction of the functions to the interval $[0, 1]$ is unessential to the theorem. To see how it may be relieved, consider a continuous monotonic transformation $x = x(t)$ defined over an interval bounded by $x_0 = x(0)$ and $x_1 = x(1)$. The inverse function $t = t(x)$ exists; and, if $f(x) = f\{t(x)\}$ and $B_n(x) = B_n\{t(x)\}$, then $B_n(x)$ converges uniformly to $f(x)$ as $B_n(t)$ converges to $f(t)$.

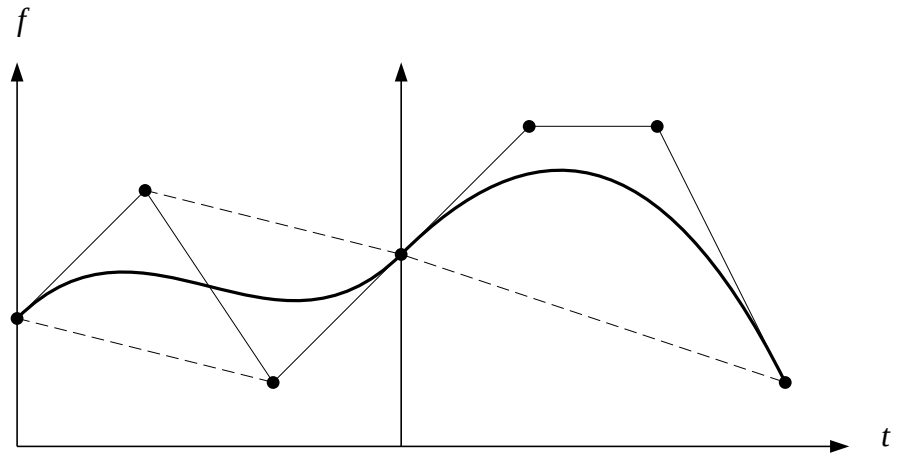

Figure 2. Adjacent cubic Bézier segments linked by a condition of first-order continuity. The solid lines link the Bezier control points in sequence. The dotted lines complete the boundaries of the convex hulls.

Whilst the Bernstein polynomials lead to an elegant constructive proof of the Weierstrass theorem, they do not in themselves provide useful polynomial approximations. One reason for their inadequacy is the slowness of their convergence to $f(t)$, which means that, in order to obtain a good approximation, a polynomial of a high degree is required. However, a high-degree polynomial is liable to have undesirable ripples. Until the advent of computer-aided design and the discoveries of Bézier, the Bernstein polynomials were regarded as little more that an adjunct to a proof of the Weierstrass theorem—see Achieser [1].

The approach of Bézier in designing a smooth curve is to use Bernstein polynomials of low degree to construct short segments which are linked by conditions of first-order continuity. An ordered set of $n+1$ points $(t_k, f_k); k = 0, \ldots, t_n$ which serves to define a Bernstein polynomial of degree n also defines a convex hull containing the path of the polynomial.

This path, which is known as the Bézier curve, has two important features. On the one hand it passes through the endpoints $(t_0, f_0), (t_n, f_n)$ which define the boundaries of a segment. This can be seen by setting $t = 0$ and $t = 1$ in (36) to give

$$B_n(0) = f_0 \qquad \text{and} \qquad B_n(1) = f_n. \tag{38}$$

On the other hand, the slopes of the vectors which are tangent to the Bézier curve at the endpoints are equal to the slopes of the adjacent sides of the polygon which forms the convex hull. Thus, maintaining assumption that the

$n+1$ points $t_0, \ldots, t_n$ are equally spaced in the interval $[0, 1]$, we have

$$\begin{aligned} B'_n(0) &= n(f_0 - f_1) & \text{and} \qquad B'_n(1) &= n(f_n - f_{n-1}) \\ &= \frac{f_0 - f_1}{t_0 - t_1} & & = \frac{f_n - f_{n-1}}{t_n - t_{n-1}}. \end{aligned} \tag{39}$$

If the endpoints of the Bézier curve are regarded as fixed, then the intermediate points $(t_1, f_1), \ldots, (t_{n-1}, f_{n-1})$ may be adjusted in an interactive manner to make the Bézier curve conform to whatever shape is desired.

Example. Consider the cubic Bernstein polynomial

$$\begin{aligned} B_3(t) &= f_0(1-t)^3 + 3f_1t(1-t)^2 + 3f_2t^2(1-t) + f_3t^3 \\ &= \alpha t^3 + \beta t^2 + \gamma t + \delta. \end{aligned} \tag{40}$$

Equating the coefficients of the powers of t shows that

$$\begin{aligned} \alpha &= f_3 - 3f_2 + 3f_1 - f_0, \\ \beta &= 3f_2 - 6f_1 + 3f_0, \\ \gamma &= 3f_1 + 3f_0, \\ \delta &= f_0. \end{aligned} \tag{41}$$

Differentiating $B_3(t)$ with respect to t gives

$$B'_3(t) = -3f_0(1-t)^2 + 3f_1t(1 - 4t + 3t^2) + 3f_2(2t - 3t^2) + 3f_3t^2, \tag{42}$$

from which

$$B'_3(0) = 3(f_0 - f_1) \qquad \text{and} \qquad B'_3(1) = 3(f_3 - f_2). \tag{43}$$

In order to exploit the Bézier command which is available in the PostScript language, we need to define the relationship between ordinates f_0, f_1, f_2, f_3 of the four control points of a cubic Bézier curve and the four parameters a, b, c, d of the representation of a cubic polynomial which is to be found under (1).

Let us imagine that the Bézier function $B_3(t)$ ranges from f_0 to f_3 as t ranges from 0 to 1, and let

$$S(x) = a(x - x_0)^3 + b(x - x_0)^2 + c(x - x_0) + d \tag{44}$$

be a segment of the cubic spline which spans the gap between two points which, for ease of notation, we will denote by (x_0, y_0) and (x_3, y_3). Then, if we define

$$\begin{aligned} x(t) &= (x_3 - x_0)t + x_0 \\ &= ht + x_0, \end{aligned} \tag{45}$$

we can identify the function $S(x)$ with the function $B(x) = B\{t(x)\}$. Thus, on taking $B(t) = \alpha t^3 + \beta t^2 + \gamma t + \delta$ and putting $t(x) = (x - x_0)/h$ in place of t, we get

$$\begin{aligned} S(x) &= \frac{\alpha}{h^3}(x - x_0)^3 + \frac{\beta}{h^2}(x - x_0)^2 + \frac{\gamma}{h}(x - x_0) + \delta \\ &= a(x - x_0)^3 + b(x - x_0)^2 + c(x - x_0) + d. \end{aligned} \tag{46}$$

The mapping from the ordinates of the Bézier control points to the parameters $\alpha, \beta, \gamma, \delta$ is given by

$$\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} ah^3 \\ bh^2 \\ ch \\ d \end{bmatrix}. \tag{47}$$

The inverse of mapping is given by

$$\begin{bmatrix} f_0, \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1/3 & 1 \\ 0 & 1/3 & 2/3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} ah^3 \\ bh^2 \\ ch \\ d \end{bmatrix}. \tag{48}$$

The PostScript Bézier command is the **curveto** command which takes as its arguments the values $x_1, f_1, x_2, f_2, x_3, f_3$ and adds a cubic Bézier segment to the current path between the current point (x_0, f_0) and the point (x_3, f_3) using (x_1, f_1) and (x_2, f_2) as the control points. Then (x_3, f_3) becomes the new current point. The **curveto** function is based upon a pair of parametric cubic equations:

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + x_0, \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + f_0. \end{aligned} \tag{49}$$

The parameters a_x, b_x, c_x are obtained from the abscissae x_0, x_1, x_2, x_3 via the transformation of (47) which is used to obtain a_y, b_y, c_y from f_0, f_1, f_2, f_3.

The parametric equation $x = x(t)$ enables the t-axis to be expanded, contracted and even folded back on itself. There is therefore no requirement that values x_0, x_1, x_2, x_3 should be equally spaced. More significantly, curves may be plotted which do not correspond to single-valued functions of x. For our own purposes, the function reduces to $x(t) = ht + x_0$ where $h = x_3 - x_0$.

The conversion of the parameters of the cubic function under (1) to the parameters of the cubic Bézier curve may be accomplished using the following procedure.

(50)
```
procedure SplinetoBezier(S : SplineVec;
                         var B : BezierVec;
                         n : integer);

  var
    i : integer;
    h, delt : real;

begin {SplinetoBezier}

  for i := 0 to n do
    begin {i}
      h := S[i + 1].x − S[i].x;
      delt := h/3;
      with B[i], S[i] do
        begin {with}
          x0 := x;
          x1 := x0 + delt;
          x2 := x1 + delt;
          x3 := x2 + delt;
          f0 := d;
          f1 := f0 + c ∗ h/3;
          f2 := f1 + (c + b ∗ h) ∗ h/3;
          f3 := f0 + (c + (b + a ∗ h) ∗ h) ∗ h
        end{with}
    end; {i}

end; {SplinetoBezier}
```

The $BezierVec$ type is defined in the following statements which must be included in the program which calls the procedure:

(51)
```
type
  BezierPoints =  record
      x0, f0, x1, f1, x2, f2, x3, f3 : real
    end;
  BezierVec = array[0..dim] of BezierPoints;
```

The Minimum-Norm Property of Splines

The draughtsman's spline assumes a shape which minimises the potential energy due to the bending strain. The strain energy is approximately proportional to the integral of the square of the second derivative along the path of

the spline; and therefore the minimisation of the potential energy leads to a property of minimum curvature. We can demonstrate that the cubic spline has a similar property; which justifies us in likening it to the draughtsman's spline.

Let $f(x) \in \mathcal{C}^2$ be any function defined over the interval $[x_0, x_n]$ which has a continuous second-order derivative. Then, as a measure of the curvature of the function, we may use the squared norm

$$\|f\|^2 = \int_{x_0}^{x_n} \big\{f''(x)\big\}^2 dx. \tag{52}$$

This differs from the ideal measure of curvature which would be the line integral of $\{f''(x)\}^2$ along the path of the function. Thus the squared norm provides only a rough approximation to the potential energy of the draughtsman's spline.

Our object is to show that, amongst all functions $f(x) \in \mathcal{C}^2$, which pass through the points $(x_i, y_i); i = 0, \ldots, n$, it is precisely the spline function which minimises the squared norm.

Let us denote the spline by $S(x)$, where $x \in [x_0, x_n]$, and let us continue to express the ith segment as

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \tag{53}$$

where $x \in [x_i, x_{i+1}]$. The derivatives of this function are

$$\begin{aligned} S_i'(x) &= 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i, \\ S_i''(x) &= 6a_i(x - x_i) + 2b_i, \\ S_i'''(x) &= 6a_i. \end{aligned} \tag{54}$$

We can establish the minimum-norm property of the cubic spline using the following result:

(55) Let $f(x) \in \mathcal{C}^2$ be a function defined on the interval $[x_0, x_n]$ which passes through the points $(x_i, y_i); i = 0, \ldots, n$ which are the knots of the spline function $S(x)$. Then

$$\|f - S\|^2 = \|f\|^2 - \|S\|^2 - 2\Big[S''(x)\big\{f'(x) - S'(x)\big\}\Big]_{x_0}^{x_n}.$$

Proof. By definition, we have

$$\begin{aligned} \|f - S\|^2 &= \|f\|^2 - 2\int_{x_0}^{x_n} f''(x)S''(x)dx + \|S\|^2 \\ &= \|f\|^2 - 2\int_{x_0}^{x_n} S''(x)\big\{f''(x) - S''(x)\big\}dx - \|S\|^2. \end{aligned} \tag{56}$$

Within this expression, we find, through integrating by parts, that

$$\int_{x_0}^{x_n} S''(x)\{f''(x) - S''(x)\}dx = \Big[S''(x)\{f'(x) - S'(x)\}\Big]_{x_0}^{x_n} - \int_{x_0}^{x_n} S'''(x)\{f'(x) - S'(x)\}dx. \tag{57}$$

Since $S(x)$ consists of the cubic segments $S_i(x); i = 0, \ldots, n-1$, it follows that the third derivative $S'''(x)$ is constant in each open interval (x_i, x_{i+1}) with a value of $S_i'''(x) = 6a_i$. Therefore

$$\begin{aligned}\int_{x_0}^{x_n} S'''(x)\{f'(x) - S'(x)\}dx &= \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} 6a_i\{f'(x) - S'(x)\}dx \\ &= \sum_{i=0}^{n-1} 6a_i\Big[f(x) - S(x)\Big]_{x_i}^{x_{i+1}} = 0,\end{aligned} \tag{58}$$

since $f(x) = S(x)$ at x_i and x_{i+1}; and hence (57) becomes

$$\int_{x_0}^{x_n} S''(x)\{f''(x) - S''(x)\}dx = \Big[S''(x)\{f'(x) - S'(x)\}\Big]_{x_0}^{x_n}. \tag{59}$$

Putting this into (56) gives the result which we wish to prove.

Now consider the case of the natural spline which satisfies the conditions $S''(x_0) = 0$ and $S''(x_n) = 0$. Putting these into the equality of (55) reduces it to

$$\|f - S\|^2 = \|f\|^2 - \|S\|^2, \tag{60}$$

which demonstrates that $\|f\|^2 \geq \|S\|^2$. In the case of a clamped spline with $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$, the equality of (55) is also reduced to that of (60). Thus we see that, in either case, the cubic spline has the minimum-norm property.

Smoothing Splines

The interpolating spline provides a useful way of approximating a smooth function $f(x) \in \mathcal{C}^2$ only when the data points lie along the path of the function or very close to it. If the data is scattered at random in the vicinity of the path, then an interpolating polynomial, which is bound to follow the same random fluctuations, will belie the nature of the underlying function. Therefore, in the interests of smoothness, we may wish to allow the spline to depart from the data points.

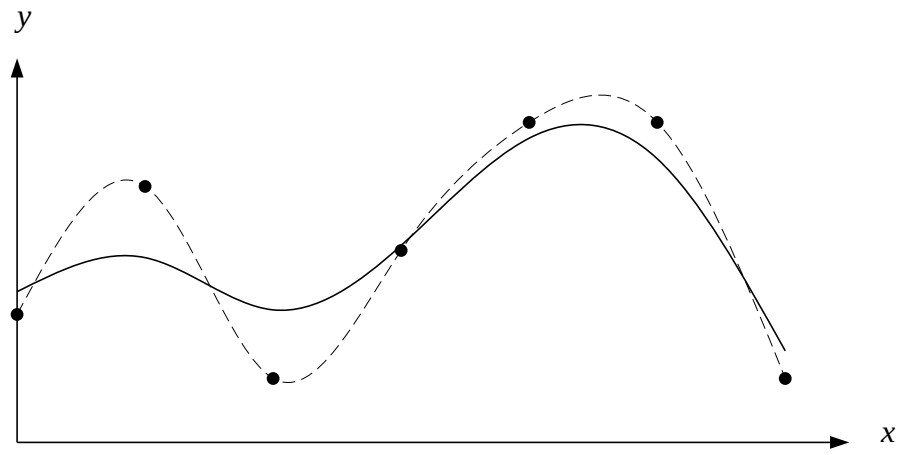

Figure 3. An cubic interpolating spline—the dotted path—and a cubic smoothing spline—the continuous path. The smoothing parameter is $\lambda = 0.5$.

We may imagine that the ordinates of the data are given by the equation

$$y_i = f(x_i) + \varepsilon_i, \tag{61}$$

where $\varepsilon_i; i = 0, \ldots, n$ form a sequence of independently distributed random variables with $V(\varepsilon_i) = \sigma_i^2$. In that case, we can attempt to reconstitute the function $f(x)$ by constructing a spline function $S(x)$ which minimises the value of

$$L = \lambda \sum_{i=0}^{n} \left(\frac{y_i - S_i}{\sigma_i} \right)^2 + (1 - \lambda) \int_{x_0}^{x_n} \left\{ S''(x) \right\}^2 dx, \tag{62}$$

wherein $S_i = S(x_i)$.

The parameter $\lambda \in [0, 1]$ reflects the relative importance which we give to the conflicting objectives of remaining close to the data, on the one hand, and of obtaining a smooth curve, on the other hand. Notice that a linear function satisfies the equation

$$\int_{x_0}^{x_n} \left\{ S''(x) \right\}^2 dx = 0, \tag{63}$$

which suggests that, in the limiting case, where $\lambda = 0$ and where smoothness is all that matters, the spline function $S(x)$ will become a straight line. At the other extreme, where $\lambda = 1$ and where the closeness of the spline to the data is the only concern, we will obtain an interpolating spline which passes exactly through the data points.

Given the piecewise nature of the spline, we can write the integral in the second term on the RHS of (62) as

$$\int_{x_0}^{x_n} \{S''(x)\}^2 dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} \{S_i''(x)\}^2 dx. \tag{64}$$

Since the spline is composed of cubic segments, the second derivative in any interval $[x_i, x_{i+1}]$ is a linear function which changes from $2b_i$ at x_i to $2b_{i+1}$ at x_{i+1}. Therefore we have

$$\begin{aligned}\int_{x_i}^{x_{i+1}} \{S_i''(x)\}^2 dx &= 4\int_0^{h_i} \left\{ b_i\left(1 - \frac{x}{h_i}\right) + b_{i+1}\frac{x}{h_i} \right\}^2 dx \\ &= \frac{4h_i}{3}\left(b_i^2 + b_i b_{i+1} + b_{i+1}^2\right),\end{aligned} \tag{65}$$

where $h_i = x_{i+1} - x_i$; and the criterion function can be rewritten as

$$L = \lambda \sum_{i=0}^{n} \left(\frac{y_i - d_i}{\sigma_i}\right)^2 + (1-\lambda)\sum_{i=0}^{n-1} \frac{4h_i}{3}\left(b_i^2 + b_i b_{i+1} + b_{i+1}^2\right), \tag{66}$$

wherein $d_i = S_i(x_i)$.

We shall treat the case of the natural spline which passes through the knots $(x_i, d_i); i = 0, \ldots, n$ and which satisfies the end conditions $S''(x_0) = 2b_0 = 0$ and $S''(x_n) = 2b_n = 0$. The additional feature of the problem of fitting a smoothing spline, compared with that of fitting an interpolating spline, is the need to determine the ordinates $d_i; i = 0, \ldots, n$ which are no longer provided by the data values $y_i; i = 0, \ldots, n$.

We can concentrate upon the problem of determining the parameters $b_i, d_i; i = 0, \ldots, n$ if we eliminate the remaining parameters $a_i, c_i; i = 1, \ldots, n-1$. Consider therefore the ith cubic segment which spans the gap between the knots (x_i, d_i) and (x_{i+1}, d_{i+1}) and which is subject to the following conditions:

$$\begin{aligned} &\text{(i)} \quad S_i(x_i) = d_i, && \text{(ii)} \quad S_i(x_{i+1}) = d_{i+1}, \\ &\text{(iii)} \quad S_i''(x_i) = 2b_i, && \text{(iv)} \quad S_i''(x_{i+1}) = 2b_{i+1}. \end{aligned} \tag{67}$$

The first condition may be regarded as an identity. The second condition, which specifies that $a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i = d_{i+1}$, gives us

$$c_i = \frac{d_{i+1} - d_i}{h_i} - a_i h_i^2 + b_i h_i. \tag{68}$$

The third condition is again an identity, whilst the fourth condition, which specifies that $2b_{i+1} = 6a_ih_i + 2b_i$, gives

$$a_i = \frac{b_{i+1} - b_i}{3h_i}. \tag{69}$$

Putting this into (68) gives

$$c_i = \frac{d_{i+1} - d_i}{h_i} - \frac{1}{3}(b_{i+1} - 2b_i)h_i. \tag{70}$$

We now have expressions for a_i and c_i which are in terms of b_{i+1}, b_i and d_{i+1}, d_i. To determine the latter parameters, we must use the conditions of first-order continuity to link the segments. The condition $S'_{i-1}(x_i) = S'_i(x_i)$ specifies that

$$3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1} = c_i. \tag{71}$$

On replacing a_{i-1} and c_{i-1} by expressions derived from (69) and (70) and rearranging the result, we get

$$b_{i-1}h_{i-1} + 2b_i(h_{i-1} + h_i) + b_{i+1}h_i = \frac{3}{h_i}(d_{i+1} - d_i) - \frac{3}{h_{i-1}}(d_i - d_{i-1}), \tag{72}$$

where $h_i = x_{i+1} - x_i$ and $h_{i-1} = x_i - x_{i-1}$. This is similar to the condition under (13). By letting i run from 1 to $n-1$ and taking account of the end conditions $b_0 = b_n = 0$, we can obtain the following matrix system,

$$\begin{bmatrix} p_1 & h_1 & 0 & \dots & 0 & 0 \\ h_1 & p_2 & h_2 & \dots & 0 & 0 \\ 0 & h_2 & p_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & p_{n-2} & h_{n-2} \\ 0 & 0 & 0 & \dots & h_{n-2} & p_{n-1} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} r_0 & f_1 & r_1 & 0 & \dots & 0 & 0 \\ 0 & r_1 & f_2 & r_2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & r_{n-2} & 0 \\ 0 & 0 & 0 & 0 & \dots & f_{n-1} & r_{n-1} \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}, \tag{73}$$

where

$$\begin{aligned} p_i &= 2(h_{i-1} + h_i), \\ r_i &= \frac{3}{h_i} \quad \text{and} \\ f_i &= -\left(\frac{3}{h_{i-1}} + \frac{3}{h_i}\right) = -(r_{i-1} + r_i). \end{aligned} \tag{74}$$

The matrix equation can be written in a summary notation as

$$Rb = Q'd. \tag{75}$$

This notation can also be used to write the criterion function of (66) as

$$L = \lambda(y - d)'\Sigma^{-1}(y - d) + \frac{2}{3}(1 - \lambda)b'Rb, \tag{76}$$

where $\Sigma = \text{diag}\{\sigma_0, \ldots, \sigma_n\}$. Using $b = R^{-1}Q'd$ enables us to write the function solely in terms of the vector d which contains the ordinates of the knots:

$$L(d) = \lambda(y - d)'\Sigma^{-1}(y - d) + \frac{2}{3}(1 - \lambda)d'QR^{-1}Q'd. \tag{77}$$

The optimal values of the ordinates are those which minimise the function $L(d)$. By differentiating with respect to d and setting the result to zero, we obtain

$$-2\lambda(y - d)'\Sigma^{-1} + \frac{4}{3}(1 - \lambda)d'QR^{-1}Q' = 0, \tag{78}$$

which is the first-order condition for minimisation. This gives

$$\begin{aligned} \lambda\Sigma^{-1}(y - d) &= \frac{2}{3}(1 - \lambda)QR^{-1}Q'd \\ &= \frac{2}{3}(1 - \lambda)Qb. \end{aligned} \tag{79}$$

When this is premultiplied by $\lambda^{-1}Q'\Sigma$ and rearranged with the further help of the identity $Rb = Q'd$, we get

$$\big(\mu Q'\Sigma Q + R\big)b = Q'y, \tag{80}$$

where $\mu = 2(1 - \lambda)/3\lambda$. Once this has been solved for b, we can obtain the value of d from equation (79). Thus

$$d = y - \mu\Sigma Qb. \tag{81}$$

The value of the criterion function is given by

$$L = (y - d)'\Sigma^{-1}(y - d) = \mu^2 b'Q'\Sigma Qb. \tag{82}$$

The matrix $A = \mu Q'\Sigma Q + R$ of equation (80) is symmetric with five diagonal bands; and we may exploit the structure of the matrix in deriving a specialised procedure for solving the equation. The procedure is as follows. First we find the factorisation $A = LDL'$ where L is a lower triangular matrix and D is a diagonal matrix. Then we take the system $LDL'b = Q'y$ in the form of $Lx = Q'y$ and we solve the latter for x which is written in place of $Q'y$. Finally, we solve $L'b = D^{-1}x$ for b which is written in place of x.

The procedure *Quincunx*, which affects this solution, takes as arguments the vectors u, v and w which are respectively the diagonal, the first supradiagonal and the second supradiagonal of the banded matrix. The vector on the LHS of the equation (80) is placed in q which contains the solution on the completion of the procedure.

(83)
```
procedure Quincunx(n : integer;
                   var u, v, w, q : nvector);

  var
    j : integer;

begin {Quincunx}

{factorisation}
  u[−1] := 0;
  u[0] := 0;
  for j := 1 to n − 1 do
    begin
      u[j] := u[j] − u[j − 2] ∗ sqr(w[j − 2]) − u[j − 1] ∗ sqr(v[j − 1]);
      v[j] := (v[j] − u[j − 1] ∗ v[j − 1] ∗ w[j − 1])/u[j];
      w[j] := w[j]/u[j];
    end;

{forward substitution}
  for j := 1 to n − 1 do
    q[j] := q[j] − v[j − 1] ∗ q[j − 1] − w[j − 2] ∗ q[j − 2];
  for j := 1 to n − 1 do
    q[j] := q[j]/u[j];

{back substitution}
  q[n + 1] := 0;
```

```
    q[n] := 0;
    for j := n − 1 downto 1 do
      q[j] := q[j] − v[j] ∗ q[j + 1] − w[j] ∗ q[j + 2];

  end; {Quincunx}
```

The procedure which calculates the smoothing spline may be envisaged as a generalisation of the procedure *CubicSplines* which calculates an interpolating spline. In fact, by setting $\lambda = 1$, we obtain the interpolating spline.

The *SmoothingSpline* procedure is wasteful of computer memory, since there is no need to store the contents of the vectors r and f which have been included in the code only for reasons of clarity. At any stage of the iteration of the index j, only two consecutive elements from each of these vectors are called for; and one of these elements may be calculated concurrently. However, the waste of memory is of little concern unless one envisages applying the procedure to a very long run of data. In that case, it should be straightforward to modify the procedure.

(84)
```
  procedure SmoothingSpline(var S : SplineVec;
                            sigma : vectors;
                            lambda : real;
                            n : integer);

    var
      h, r, f, p, q, u, v, w : vectors;
      i, j : integer;
      mu : real;

  begin {SmoothingSpline}
    mu := 2 ∗ (1 − lambda)/(3 ∗ lambda);

    h[0] := S[1].x − S[0].x;
    r[0] := 3/h[0];
    for i := 1 to n − 1 do
      begin
        h[i] := S[i + 1].x − S[i].x;
        r[i] := 3/h[i];
        f[i] := −(r[i − 1] + r[i]);
        p[i] := 2 ∗ (S[i + 1].x − S[i − 1].x);
        q[i] := 3 ∗ (S[i + 1].y − S[i].y)/h[i] − 3 ∗ (S[i].y − S[i − 1].y)/h[i − 1];
      end;

    for i := 1 to n − 1 do
```

```
      begin
        u[i] := sqr(r[i − 1]) ∗ sigma[i − 1]
              +sqr(f[i]) ∗ sigma[i] + sqr(r[i]) ∗ sigma[i + 1];
        u[i] := mu ∗ u[i] + p[i];
        v[i] := f[i] ∗ r[i] ∗ sigma[i] + r[i] ∗ f[i + 1] ∗ sigma[i + 1];
        v[i] := mu ∗ v[i] + h[i];
        w[i] := mu ∗ r[i] ∗ r[i + 1] ∗ sigma[i + 1];
      end;

    Quincunx(n, u, v, w, q);

  {Spline Parameters}
    S[0].d := S[0].y − mu ∗ r[0] ∗ q[1] ∗ sigma[0];
    S[1].d := S[1].y − mu ∗ (f[1] ∗ q[1] + r[1] ∗ q[2]) ∗ sigma[0];
    S[0].a := q[1]/(3 ∗ h[0]);
    S[0].b := 0;
    S[0].c := (S[1].d − S[0].d)/h[0] − q[1] ∗ h[0]/3;
    r[0] := 0;

    for j := 1 to n − 1 do
      begin
        S[j].a := (q[j + 1] − q[j])/(3 ∗ h[j]);
        S[j].b := q[j];
        S[j].c := (q[j] + q[j − 1]) ∗ h[j − 1] + S[j − 1].c;
        S[j].d := r[j − 1] ∗ q[j − 1] + f[j] ∗ q[j] + r[j] ∗ q[j + 1];
        S[j].d := y[j] − mu ∗ S[j].d ∗ sigma[j];
      end;

  end; {SmoothingSpline}
```

A Stochastic Model for the Smoothing Spline

The disadvantage of the smoothing spline is the extent to which the choice of the value for the smoothing parameter remains a matter of judgment. One way of avoiding such judgments is to adopt an appropriate model of the process which has generated the data to which the spline is to be fitted. Then the value of the smoothing parameter may be determined in the process of fitting the model.

Since the smoothing spline is a continuous function, it is natural to imagine that the process underlying the data is also continuous. A model which is likely to prove appropriate to many circumstances is a so-called integrated Wiener process which is the continuous analogue of the familiar discrete-time unit-root autoregressive processes. To the continuous process, a discrete process is

added which represents a set of random errors of observation. Therefore, the estimation of the trend becomes a matter of signal extraction. A Wiener process $Z(t)$ consists of an accumulation of independently distributed stochastic increments. The path of $Z(t)$ is continuous almost everywhere and differentiable almost nowhere. If $dZ(t)$ stands for the increment of the process in the infinitesimal interval dt, and if $Z(a)$ is the value of the function at time a, then the value at time $\tau > a$ is given by

$$Z(\tau) = Z(a) + \int_a^\tau dZ(t). \tag{85}$$

Moreover, it is assumed that the change in the value of the function over any finite interval $(a, \tau]$ is a random variable with a zero expectation:

$$E\big\{Z(\tau) - Z(a)\big\} = 0. \tag{86}$$

Let us write $ds \cap dt = \emptyset$ whenever ds and dt represent non-overlapping intervals. Then the conditions affecting the increments may be expressed by writing

$$E\big\{dZ(s)dZ(t)\big\} = \begin{cases} 0, & \text{if } ds \cap dt = \emptyset; \\ \sigma^2 dt, & \text{if } ds = dt. \end{cases} \tag{87}$$

These conditions imply that the variance of the change over the interval $(a, \tau]$ is proportional to the length of the interval. Thus

$$\begin{aligned} V\big\{Z(\tau) - Z(a)\big\} &= \int_{s=a}^{\tau} \int_{t=a}^{\tau} E\big\{dZ(s)dZ(t)\big\} \\ &= \int_{t=a}^{\tau} \sigma^2 dt = \sigma^2(\tau - a). \end{aligned} \tag{88}$$

The definite integrals of the Wiener process may be defined also in terms of the increments. The value of the first integral at time τ is given by

$$\begin{aligned} Z^{(1)}(\tau) &= Z^{(1)}(a) + \int_a^\tau Z(t)dt \\ &= Z^{(1)}(a) + Z(a)(\tau - a) + \int_a^\tau (\tau - t)dZ(t), \end{aligned} \tag{89}$$

where the second equality comes via (85). The mth integral is

$$Z^{(m)}(\tau) = \sum_{k=0}^{m} Z^{(k-m)}(a)\frac{(\tau - a)^k}{k!} + \int_a^\tau \frac{(\tau - t)^m}{m!}dZ(t). \tag{90}$$

The covariance of the changes $Z^{(j)}(\tau) - Z^{(j)}(a)$ and $Z^{(k)}(\tau) - Z^{(k)}(a)$ of the jth and the kth integrated processes derived from $Z(t)$ is given by

$$\begin{aligned} C_{(a,\tau)}\{z^{(j)}, z^{(k)}\} &= \int_{s=a}^{\tau}\int_{t=a}^{\tau} \frac{(\tau-s)^j(\tau-t)^k}{j!k!} E\{dZ(s)dZ(t)\} \\ &= \sigma^2 \int_a^{\tau} \frac{(\tau-t)^j(\tau-t)^k}{j!k!}dt = \sigma^2 \frac{(\tau-a)^{j+k+1}}{(j+k+1)!j!k!}. \end{aligned} \tag{91}$$

The simplest stochastic model which can give rise to the smoothing spline is one in which the generic observation is depicted as the sum of a tend component described by an integrated Wiener process and a random error taken from a discrete white-noise sequence. We may imagine that the observations $y_0, y_1, \ldots, y_n$ are made at the times $t_0, t_1, \ldots, t_n$. The interval between t_{i+1} and t_i is $h_i = t_{i+1} - t_i$ which, for the sake of generality, is allowed to vary. These points in time replace the abscissae $x_0, x_1, \ldots, x_n$ which have, hitherto, formed part of our observations.

In order to conform to our existing notation, we define

$$c_i = Z(t_i) \qquad \text{and} \qquad d_i = Z^{(1)}(t_i) \tag{92}$$

to be, respectively, the slope of the trend component and its level at time t_i, where $Z(t_i)$ and $Z^{(1)}(t_i)$ are described by equations (85) and (89). Also we define

$$\zeta_{i+1} = \int_{t_i}^{t_{i+1}} dZ(t) \qquad \text{and} \qquad \nu_{i+1} = \int_{t_i}^{t_{i+1}} (t_{i+1} - t)dZ(t). \tag{93}$$

Then the model of the underlying trend can be written is state-space form as follows:

$$\begin{bmatrix} d_{i+1} \\ c_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & h_i \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_i \\ c_i \end{bmatrix} + \begin{bmatrix} \nu_{i+1} \\ \zeta_{i+1} \end{bmatrix}, \tag{94}$$

whilst the equation of the corresponding observation is

$$y_{i+1} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} d_{i+1} \\ c_{i+1} \end{bmatrix} + \varepsilon_{i+1}. \tag{95}$$

Using the result under (92), we find that the dispersion matrix for the state disturbances is

$$D \begin{bmatrix} \nu_{i+1} \\ \zeta_{i+1} \end{bmatrix} = \sigma_\varepsilon^2 \phi \begin{bmatrix} \frac{1}{3}h_i^3 & \frac{1}{2}h_i^2 \\ \frac{1}{2}h_i^2 & h_i \end{bmatrix}, \tag{96}$$

where $\sigma_\varepsilon^2\phi = \sigma_\zeta^2$ is the variance of the Wiener process expressed as the product of the variance σ_ε^2 of the observations errors and of the signal-to-noise ratio $\phi = \sigma_\zeta^2/\sigma_\varepsilon^2$.

The estimation of the model according to the criterion of maximum likelihood is accomplished by a straightforward application of the Kalman filter which serves to generate the prediction errors whose sum of squares is the major element of the criterion function. In fact, when it has has been concentrated in respect of σ_ε^2, the criterion function has the signal-to-noise ratio ϕ as its sole argument. Once the minimising value of ϕ has been determined, the definitive smoothed estimates of the state parameters c_i, d_i for $i = 0, \ldots, n$ may be obtained via one of the algorithms presented in an account by Merkus et al. [7].

In order to estimate the path of the trend, it is necessary to represent the values which lie between the adjacent points (t_i, d_i), (t_{i+1}, d_{i+1}) by an interpolated function whose first derivatives at the two points are given by c_i and c_{i+1}. It has been demonstrated by Wahba [13] that the curve which represents the minimum-mean-square-error estimator of a trend generated by an integrated Wiener process is a smoothing spline. The practical details of constructing the spline have been set forth by Wecker and Ansley [11]. A lucid exposition, which we shall follow here, has been provided recently by de Vos and Steyn [10].

The problem of estimating the intermediate value of the trend between the times t_i and t_{i+1} of two adjacent observations is that of finding its expectation conditional upon the values $\xi_i = (c_i, d_i)$ and $\xi_{i+1} = (c_{i+1}, d_{i+1})$. Let $t \in (t_i, t_{i+1}]$ be the date of the intermediate values c_t and d_t; and let us define the following quantities which represent the stochastic increments which accumulate over the sub-intervals $(t_i, t]$ and $(t, t_{i+1}]$:

$$\begin{aligned} \zeta_t &= \int_{t_1}^{t} dZ(\tau), & \bar{\zeta}_t &= \int_{t}^{t_{i+1}} dZ(\tau), \\ \nu_t &= \int_{t_1}^{t} (t - t_i) dZ(\tau), & \bar{\nu}_t &= \int_{t}^{t_{i+1}} (t_{i+1} - t) dZ(\tau). \end{aligned} \tag{97}$$

In these terms, the stochastic increments over the entire interval $(t_i, t_{i+1}]$ are given by

$$\begin{bmatrix} \nu_{i+1} \\ \zeta_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & (t_{i+1} - t) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu_t \\ \zeta_t \end{bmatrix} + \begin{bmatrix} \bar{\nu}_t \\ \bar{\zeta}_t \end{bmatrix}, \tag{98}$$

which is a variant of equation (94).

The values of the slope and the level of the Wiener process at time t can

be given in terms of two of the quantities under (97) as follows:

$$\begin{aligned} c_t &= c_i + \zeta_t \quad \text{and} \\ d_t &= d_i + (t - t_i)c_i + \nu_t. \end{aligned} \tag{99}$$

After the rest of the interval from t to t_{i+1} has been covered, the slope and the level become

$$\begin{aligned} c_{i+1} &= c_t + \bar{\zeta}_t \quad \text{and} \\ d_{i+1} &= d_t + (t_{i+1} - t)c_t + \bar{\nu}_t, \end{aligned} \tag{100}$$

which entail the remaining quantities under (97). Substituting for c_t and d_t in these expressions gives

$$\begin{aligned} c_{i+1} &= c_i + \zeta_t + \bar{\zeta}_t \quad \text{and} \\ d_{i+1} &= d_i + h_i c_i + (t_{i+1} - t)\zeta_t + \nu_t + \bar{\nu}_t, \end{aligned} \tag{101}$$

wherein $(t_{i+1} - t)\zeta_t + \nu_t + \bar{\nu}_t = \nu_{i+1}$ is an expression which is also provided by equation (98).

The equations of (99) and (101) enable us to evaluate the joint moments of d_t, d_{i+1} and c_{i+1} conditional upon the values c_i and d_i. Thus, with reference to the result under (91), we find that

$$C(d_t, c_{i+1}) = C(\nu_t, \zeta_t) = \frac{1}{2}(t - t_i)^2 \tag{102}$$

and that

$$\begin{aligned} C(d_t, d_{i+1}) &= (t_{i+1} - t)C(\nu_t, \zeta_t) + V(\nu_t) \\ &= \frac{1}{2}(t_{i+1} - t)(t - t_i)^2 + \frac{1}{3}(t - t_i)^3. \end{aligned} \tag{103}$$

The conditional expectation of the intermediate trend value d_t is given by the regression equation

$$E\big(d_t\big|\mathcal{I}_{i+1}\big) = E\big(d_t\big|\mathcal{I}_i\big) + C\big(d_t, \xi_{i+1}\big)D\big(\xi_{i+1}\big)^{-1}\big(\xi_{i+1} - E\big\{\xi_{i+1}|\mathcal{I}_i\big\}\big), \tag{104}$$

where $\xi_{i+1} = (d_{i+1}, c_{i+1})$, and where $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$ represent the information available at t_i and t_{i+1} which is conveyed, in fact, by the values of ξ_i and ξ_{i+1}.

On the RHS of the expression there is

$$\begin{aligned} E\big(d_t\big|\mathcal{I}_i\big) &= d_i + (t - t_i)c_i \quad \text{and} \\ \xi_{i+1} - E\big\{\xi_{i+1}|\mathcal{I}_i\big\} &= \begin{bmatrix} d_{i+1} - d_i - h_i c_i \\ c_{i+1} - c_i \end{bmatrix}. \end{aligned} \tag{105}$$

Of the remaining terms on the RHS, the elements of the vector $C(d_t, \xi_{i+1}) = [C(d_t, d_{i+1}), C(d_t, c_{i+1})]$ are found under (102) and (103), whilst the dispersion matrix $D(\xi_{i+1}) = D[\nu_{i+1}, \zeta_{i+1}]$ is to be found under (96).

Detailed computation shows that the regression equation is a cubic function of t of the form

$$f(t) = a_i(t - t_i)^3 + b_i(t - t_i)^2 + c_i(t - t_i) + d_i \tag{106}$$

wherein

$$a_i = \frac{1}{h_i^2}(c_i + c_{i+1}) + \frac{2}{h_i^3}(d_i - d_{i+1}) \tag{107}$$

and

$$b_i = \frac{3}{h_i^2}(d_{i+1} - d_i) - \frac{1}{h_i}(c_{i+1} + 2c_i). \tag{108}$$

The expressions for a_i and b_i could be obtained from those under (22) and (23) simply by substituting d_{i+1} and d_i for y_{i+1} and y_i respectively. The latter expressions relate to an segment of an interpolating spline of which the ends have been clamped.

The mere fact that the estimate of the stochastic trend between the points (t_i, d_i) and (t_{i+1}, d_{i+1}) has the same form as a segment of a spline does not establish that the estimated trend function as a whole is equivalent to a smoothing spline. Some further results are needed. First it has to be demonstrate that the condition of second-order continuity is satisfied at the junctures of adjacent segments of the estimated trend. Then it has to be show that the knots of the segmented trend curve are identical to those which would be generated by a smoothing spline subject to a particular value for the smoothing parameter. A demonstration of these results, which is on an abstract level, has been provided by Wahba [13].

References

[1] Achieser, N.I., (1956), Theory of Approximation, Fredrick Ungar Publishing Co, New York.

[2] Adobe Systems Inc., (1985), The PostScript Language Reference Manual, Addison–Wesley Publishing Co., Reading Mass.

[3] Bernstein, S. N. (1912), "Démonstration du Théorème de Weierstrass Fondée sur le Calcul des Probabilitiés", Proceedings of the Mathematical Society of Kharkov, 13, 1-2.

[4] Bézier, P., (1966), "Définition Numérique des Courbes et Surfaces I", Automatisme, 11, 625–632.

[5] Bézier, P., (1967), "Définition Numérique des Courbes et Surfaces II", Automatisme", 12, 17–21.

[6] Craven, P. and Grace Wahba, (1979), "Smoothing Noisy Data with Spline Functions: Estimating the Correct Degree of Smoothing by the Method of Generalised Cross-Validation", Numerische Mathematik, 31, 377–403.

[7] Merkus, H.R., D.S.G. Pollock and A.F. de Vos, (1991) "A Synopsis of the Smoothing Formulae Associated with the Kalman Filter", Discussion paper No. 246 of the Department of Economics of Queen Mary College, forthcoming in Computer Science in Economics and Management.

[8] Reinsch, (1967),"Smoothing by Spline Functions", Numerische Mathematik, 10, 177–183.

[9] Schoenberg, (1964), "Spline Functions and the Problem of Graduation", Proc. Nat. Acad. Sci., 52, 947–950.

[10] De Vos, A.F. and I.J. Steyn, (1990), "Stochastic Nonlinearity: A Firm Basis for the Flexible Functional Form", Research Memorandum 1990-13, Vrije Universiteit Amsterdam.

[11] Wecker, W.P. and C.F. Ansley, (1983),"The Signal Extraction Approach to Nonlinear Regression and Spline Smoothing",Journal of the American Statistical Association, 78, 81–89.

[12] Weierstrass, K., (1885), "über die analytische Darstellbarkheit sogenannter willhülicher Functionen einer reellen Veränderlichen", Berliner Berichte.

[13] Wahba, Grace, (1987), "Improper Priors, Spline Smoothing and the Problem of Guarding against Model Errors in Regression", Journal of the Royal Statistical Society, Series B . 40, 364–372.

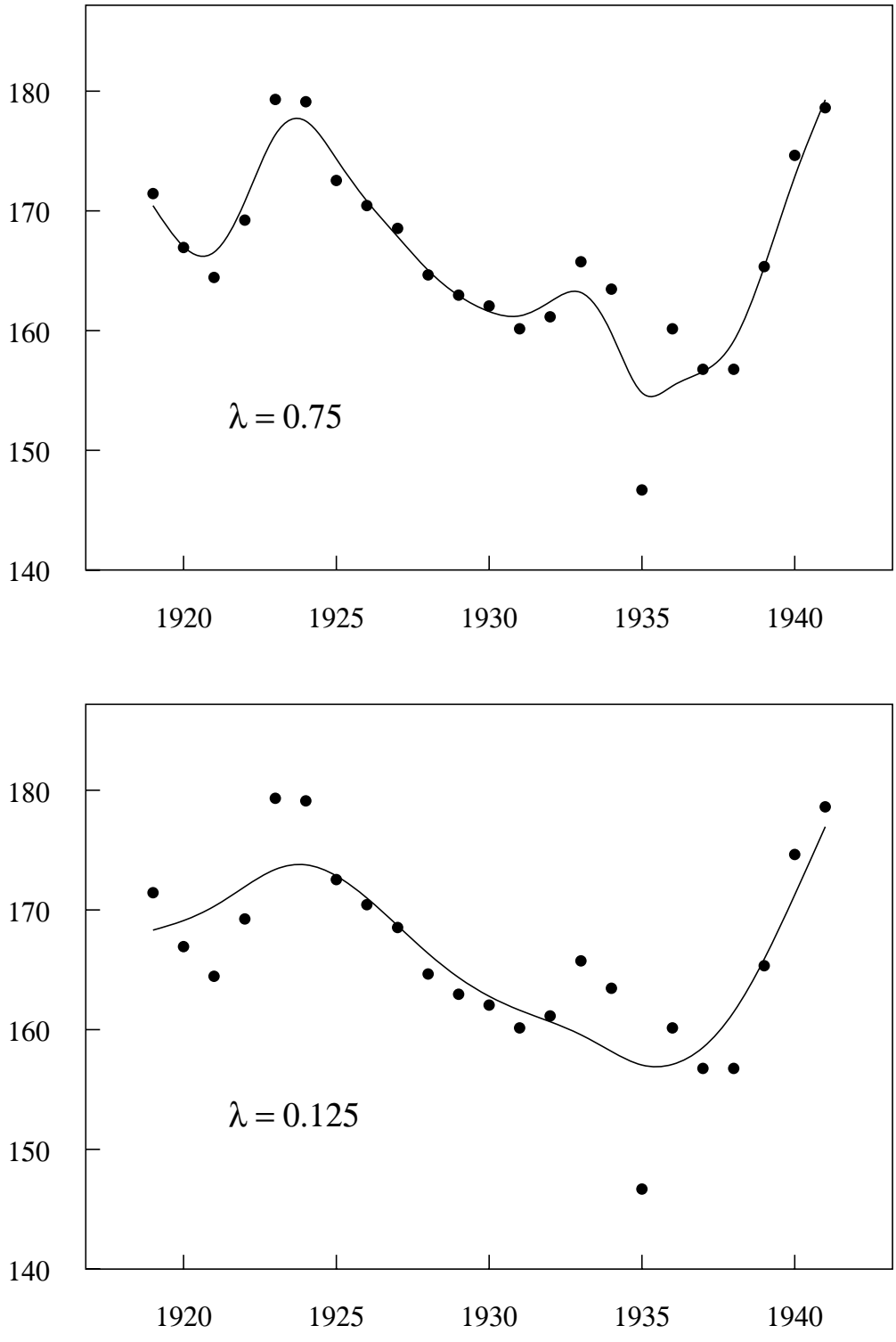

Figure 4. Cubic smoothing splines fitted to data on meat consumption in the United States, 1919–1941.

SMOOTHING WITH CUBIC SPLINES

by

D.S.G. Pollock

Queen Mary and Westfield College,
The University of London

This paper presents the algorithms of the cubic interpolating spline and the smoothing spline together with their implementations in Pascal.

The smoothing spline can be used for estimating trends in time series. The Pascal code makes use of the facility for constructing cubic Bézier curves which is available in the PostScript graphics language.

The interpolating spline is controlled by a single parameter which governs the trade-off between the smoothness of the curve and its closeness to the data points. Users must rely on their own judgment in choosing a value for this parameter.

Some statisticians have voiced concern about placing such reliance upon the user's judgment; and various criteria have been proposed for determining the degree of smoothness automatically. In the final section of the paper, there is a brief account of the model-based approach to spline smoothing which has emerged in recent years and which is still being developed.

Address for correspondence:

D.S.G. Pollock
Department of Economics
Queen Mary College
University of London
Mile End Road
London E1 4 NS

Tel : +44-71-975-5096
Fax : +44-71-975-5500