Shape-preserving Knot Removal

Larry L. Schumaker † and Sonya S. Stanley ‡

Dedicated to John Gregory

Abstract. Starting with a shape-preserving C^1 quadratic spline, we show how knots can be removed to produce a new spline which is within a specified tolerance of the original one, and which has the same shape properties. We give specific algorithms and some numerical examples, and also show how the method can be used to compute approximate best free-knot splines. Finally, we discuss how to handle noisy data, and develop an analogous knot removal algorithm for a monotonicity preserving surface method.

Keywords. Shape-preserving spline, knot removal, monotone surfaces

§1. Introduction

The idea of removing knots from a spline function in order to produce a good approximation with fewer parameters has been discussed in a number of recent papers [Lyche & Mørken '87a, '87b, '88, Arge et al '90, Lyche '92]. See also the book [Goldman & Lyche '93]. Starting with a given B-spline expansion f, these authors construct another B-spline expansion g with fewer knots which differs from f by less than some given tolerance. The method in [Lyche & Mørken '87a, '87b, '88, Lyche '92] uses a discrete least squares approximation process to construct g, and to decide which knots should be removed.

In many practical applications, it is important to construct a spline whose shape accurately models the shape of the input data. In [Arge et al '90], the authors extended the knot removal method to enforce constraints such as positivity, monotonicity, and convexity. Their method requires repeated solutions of quadratic minimization problems with linear constraints.

In the first part of this paper we consider knot removal based on the shape-preserving C^1 quadratic interpolating splines discussed in [McAllister & Roulier '81, Schumaker '83, DeVore & Yan '86]. We believe this approach has several advantages:

1) shape-preservation is built into the algorithm,

2) in contrast to B-spline based methods, our algorithm is completely local,

† Department of Mathematics, Vanderbilt University, Nashville, TN 37240, s@mars.cas.vanderbilt.edu. Supported by the National Science Foundation under grant DMS-9208413.

‡ Department of Mathematics, Vanderbilt University, Nashville, TN 37240, sstanley@math.vanderbilt.edu.

3) the error is measured in terms of norms on the functions themselves rather than equivalent coefficient norms,

4) our algorithm is more efficient since it does not require solving optimization problems.

The details of our method are developed in Sections 2,3 and 4, while numerical examples can be found in Section 5. In Section 6 we discuss how to use our knot removal algorithm to find good knots for approximating given functions by splines. In Section 7 we show how knot removal can also be used in the fitting of noisy data.

In [Lyche & Mørken '87a] an algorithm was developed for removing knot (lines) from tensor-product B-spline surfaces. The algorithm is not designed to preserve the shape of the resulting surfaces. In the second part of this paper (Sections 8 and 9), we suggest a different approach to constructing approximations to surfaces based on certain C^1 cubic interpolating splines developed in [Han & Schumaker '95] for fitting monotone surfaces. When applied to monotone data, this approach allows knot line removal while preserving the monotonicity of the resulting surfaces. We conclude the paper with a collection of remarks.

§2. Removing Knots from a C^1 Quadratic Spline

Suppose that $\mathcal{S}_2^1(\Delta)$ denotes the space of C^1 continuous quadratic splines defined on an interval $[a,b]$ with knots $a=\tau_1<\tau_2<\cdots<\tau_N=b$. In this section we describe an algorithm for removing knots from a quadratic C^1 interpolating spline s without perturbing the spline more than a given tolerance. At each step, we replace a pair of neighboring knots with one new one, so that at each step we get a new spline with one less quadratic polynomial piece. This requires that we calculate new coefficients for two quadratic segments. We now describe this process in more detail. The method is based on the following lemma of [Schumaker'83].

Lemma 2.1. *Let $[t_1,t_2]$ be a fixed interval, and let ξ lie in its interior. Suppose z_i and s_i are prescribed for $i=1,2$. Then there exists a C^1 quadratic spline g defined on $[t_1,t_2]$ with one knot at ξ such that*

$$g(t_i)=z_i, \quad g'(t_i)=s_i, \qquad i=1,2.$$

Proof: Let

$$g(t)=\begin{cases} A_1+B_1(t-t_1)+C_1(t-t_1)^2, & \text{if} \quad t_1\le t<\xi, \\ \tilde{A}_1+\tilde{B}_1(t-\xi)+\tilde{C}_1(t-\xi)^2, & \text{if} \quad \xi\le t\le t_2, \end{cases} \tag{2.1}$$

where

$$A_1=z_1, \quad B_1=s_1, \quad C_1=(\bar{s}-s_1)/2\alpha \tag{2.2}$$

$$\tilde{A}_1=A_1+B_1\alpha+C_1\alpha^2, \quad \tilde{B}_1=\bar{s}, \quad \tilde{C}_1=(s_2-\bar{s})/2\beta, \tag{2.3}$$

$$\bar{s}=\frac{2(z_2-z_1)-(\alpha s_1+\beta s_2)}{(t_2-t_1)}, \qquad \alpha=\xi-t_1, \quad \beta=t_2-\xi. \tag{2.4}$$

It is easy to see (cf. [Schumaker'83]) that g satisfies the desired interpolation conditions. ■

To apply this lemma to a spline $s \in S_2^1(\Delta)$, we choose $t_1 = \tau_j$, $z_1 = s(\tau_j)$, $s_1 = s'(\tau_j)$, and $t_2 = \tau_{j+3}$, $z_2 = s(\tau_{j+3})$, and $s_2 = s'(\tau_{j+3})$. Then clearly

$$s_j(t) = \begin{cases} g(t), & t \in [\tau_j, \tau_{j+3}], \\ s(t), & \text{otherwise} \end{cases}$$

is a C^1 quadratic spline in $\mathcal{S}_2^1(\Delta_j)$, where Δ_j is obtained from Δ by replacing τ_{j+1} and τ_{j+2} by the single knot ξ.

The interval $J_j = [\tau_j, \tau_{j+3}]$ can be any of the intervals $[\tau_1, \tau_4]$, ..., $[\tau_{N-3}, \tau_N]$. To decide which one to use, we assign a weight to each of the J_j which measures the difference between the initial spline s and the spline s_j which would arise if we replaced the knots τ_{j+1} and τ_{j+2} by a single knot, i.e.,

$$w_j = \|s - s_j\|, \qquad j = 1, \ldots N-3.$$

Then if we choose ν such that $w_\nu = \min\{w_j\}$, the new spline $\tilde{s} = s_\nu$ will be the one which differs the least from s.

This process can be repeated recursively. The following algorithm starts with an initial spline $s^{(0)}$, and computes a sequence of splines, each with one fewer knot. It stops when the difference between the next approximation $s^{(k)}$ and the initial spline $s^{(0)}$ becomes larger than a prescribed tolerance.

Algorithm 2.2. *Let $tol > 0$, and let $s^{(0)}$ be a given C^1 quadratic spline on an interval $[a, b]$ with knots $\tau_i^{(0)} = \tau_i$, $i = 1, \ldots, N$. Let $k = 1$.*

1) *for $j = 1, \ldots, N-k-2$, compute $w_j^{(k)} = \|s^{(0)} - s_j^{(k)}\|$, where $s_j^{(k)}$ is the spline obtained from $s^{(k-1)}$ by removing the knots $\tau_{j+1}^{(k-1)}$, $\tau_{j+2}^{(k-1)}$ from the interval $J_j^{(k-1)} = [\tau_j^{(k-1)}, \tau_{j+3}^{(k-1)}]$ and replacing them by one new knot $\xi_j^{(k)}$ in $J_j^{(k-1)}$,*
2) *find ν such that $w_\nu = \min\{w_j^{(k)}\}$,*
3) *if $w_\nu > tol$*
   stop
   else
   $s^{(k)} := s_j^{(k)}$
   $k = k + 1$
   return to 1).
   endif

Discussion: The spline $s^{(k)}$ has knots

$$\tau_i^{(k)} = \begin{cases} \tau_i^{(k-1)}, & i = 1, \ldots, \nu, \\ \xi_\nu^{(k)}, & i = \nu + 1, \\ \tau_{i+1}^{(k-1)}, & i = \nu + 2, \ldots, N - k. \end{cases}$$

The norm used in 1) can be any of the usual discrete norms. The examples presented in Sect. 5 below are based on the discrete uniform norm, which is calculated by finding the maximum value on a fine mesh of equally spaced points in $[a, b]$. In calculating the weight $w_j^{(k)}$ in step 1), for $k > 1$ it suffices to look only at points in the subinterval $[\tau_j^{(k-1)}, \tau_{j+3}^{(k-1)}]$ since $s_j^{(k)}$ differs from $s^{(k-1)}$ only in this interval. Moreover, for $k > 1$ the only weights that have to be calculated are $w_j^{(k)}$ for $j = \nu - 2, \ldots, \nu + 1$ since

$$w_i^{(k)} = \begin{cases} w_i^{(k-1)}, & i = 1, \ldots, \nu - 3, \\ w_{i+1}^{(k-1)}, & i = \nu + 2, \ldots, N - k - 3. \end{cases}$$

This algorithm will work with any choice of $\xi_j^{(k)}$ in the intervals $J_j^{(k-1)}$, but they have to be chosen carefully if we want to preserve the shape of the initial spline $s^{(0)}$. We discuss how to do this in Section 4 below. In implementing this algorithm, we have found it convenient to simply maintain a list of which knots remain in the model, rather than shifting and renumbering all of the knot and coefficient information. ■

§3. Computing an Initial Spline

Suppose we are given data $\{(t_i, z_i)\}_{i=1}^n$. To apply Algorithm 2.2, we first need to construct an initial fit using C^1 quadratic splines. Since we are interested in shape, it is natural to use the shape-preserving method discussed in [McAllister & Roulier '81, Schumaker '83, DeVore & Yan '86]. Since we need the notation later, we give a brief review, based on the general treatment in [Schumaker '83].

Applying Lemma 2.1 to each subinterval $[t_i, t_{i+1}]$ of $[t_1, t_n]$ produces a spline

$$s(t) = \begin{cases} A_i + B_i(t - t_i) + C_i(t - t_i)^2, & \text{if} \quad t_i \le t < \xi_i, \\ \tilde{A}_i + \tilde{B}_i(t - \xi_i) + \tilde{C}_i(t - \xi_i)^2, & \text{if} \quad \xi_i \le t \le t_{i+1} \end{cases} \tag{3.1}$$

having knots

$$\{t_1 < \xi_1 < t_2 < \xi_2 < t_3 < \cdots < t_{n-1} < \xi_{n-1} < t_n\}.$$

It was first shown in [McAllister & Roulier '81] that with appropriate choices of the $\xi_i \in (t_i, t_{i+1})$ and the slopes $s_i = s'(t_i)$, splines of this type are capable of preserving monotonicity and convexity properties of the data $\{(t_i, z_i)\}_{i=1}^n$. We now state some relevant results.

Definition 3.1. *Let $1 \le i \le n - 1$, and let*

$$\delta_i = \frac{z_{i+1} - z_i}{t_{i+1} - t_i}. \tag{3.2}$$

We say that the data for the interval $I_i = [t_i, t_{i+1}]$ are M-consistent provided that one of the following holds:

1) $s_i = s_{i+1} = \delta_i = 0$;

2) $\delta_i \neq 0$ *and* $s_i, s_{i+1}, \delta_i \leq 0$;

3) $\delta_i \neq 0$ *and* $s_i, s_{i+1}, \delta_i \geq 0$.

It is easy to see that if for some interval I_i the data are not M-consistent, then the spline s cannot be monotone on that interval. We want to construct a spline s as in (3.1) which is monotone on all intervals I_i where M-consistency holds. It was shown in [Schumaker '83] that, in general, for a given interval I_i, having M-consistent data is not enough to insure that the the spline s is monotone on I_i. To insure monotonicity, it may also be necessary to enforce a condition on the relationship between the size of the slopes s_i and s_{i+1} and the location of the point ξ_i.

Lemma 3.2. *For each* $i = 1, \ldots, n-1$, *let* $J_i = (t_i, t_{i+1})$, *and define*

$$I_i^M = \begin{cases} (t_i, \bar{\xi}_i] \cap J_i, & \text{if } \delta_i, s_i, s_{i+1} \geq 0 \text{ and } s_i > s_{i+1} \\ [\bar{\xi}_i, t_{i+1}) \cap J_i, & \text{if } \delta_i, s_i, s_{i+1} \geq 0 \text{ and } s_i < s_{i+1} \\ [\bar{\xi}_i, t_{i+1}) \cap J_i, & \text{if } \delta_i, s_i, s_{i+1} \leq 0 \text{ and } s_i > s_{i+1} \\ (t_i, \bar{\xi}_i] \cap J_i, & \text{if } \delta_i, s_i, s_{i+1} \leq 0 \text{ and } s_i < s_{i+1} \\ J_i, & \text{otherwise.} \end{cases}$$

where

$$\bar{\xi}_i = t_i + (t_{i+1} - t_i)(2\delta_i - s_{i+1})/(s_i - s_{i+1}).$$

Suppose that the data are M-consistent on the interval $I_i = [t_i, t_{i+1}]$. *Then the spline* s *defined in (3.1) is monotone on* I_i *if and only if* ξ_i *is chosen in* I_i^M.

Proof: The condition that $\xi_i \in I_i^M$ is equivalent to conditions (2.7) and (2.8) of Lemma 2.5 in [Schumaker '83]. It was shown there that those conditions are equivalent to s being monotone on I_i when the data are M-consistent on I_i. ■

Concerning convexity and concavity, we have the following analog of Definition 3.1:

Definition 3.3. *We say that the data for the interval* $I_i = [t_i, t_{i+1}]$ *are C-consistent provided that one of the following holds:*

1) $s_i = s_{i+1} = \delta_i$;

2) $(s_{i+1} - \delta_i)(s_i - \delta_i) < 0$.

If $s_{i+1} - \delta_i$ and $s_i - \delta_i$ have the same sign and at least one of them is nonzero, then clearly s must have an inflection point in $[t_i, t_{i+1}]$. Thus, if the data are not C-consistent for an interval I_i, there cannot exist s which is convex or concave on that interval.

As with monotonicity, we want to construct a spline (3.1) which is convex or concave on intervals where the data are C-consistent. In case 1) of Definition 3.3, this is easy,

since then we can choose s to be the linear polynomial joining the data points (t_i, z_i) and (t_{i+1}, z_{i+1}). The following lemma shows what happens for other values of the slopes s_i and s_{i+1}.

Lemma 3.4. *For each $i = 1, \ldots, n-1$, let*

$$I_i^C = \begin{cases} (t_i, \bar{t}_i] \cap J_i, & \text{if } (s_{i+1} - \delta_i)(s_i - \delta_i) < 0 \text{ and } |s_{i+1} - \delta_i| < |s_i - \delta_i| \\ [\tilde{t}_i, t_{i+1}) \cap J_i, & \text{if } (s_{i+1} - \delta_i)(s_i - \delta_i) < 0 \text{ and } |s_{i+1} - \delta_i| > |s_i - \delta_i| \\ I_i^M, & \text{otherwise,} \end{cases}$$

where $J_i = (t_i, t_{i+1})$, and

$$\begin{aligned} \bar{t}_i &= t_i + 2(t_{i+1} - t_i)(s_{i+1} - \delta_i)/(s_{i+1} - s_i) \\ \tilde{t}_i &= t_{i+1} + 2(t_{i+1} - t_i)(s_i - \delta_i)/(s_{i+1} - s_i). \end{aligned}$$

Suppose that the data are C-consistent on the interval $I_i = [t_i, t_{i+1}]$. Then the spline s defined in (3.1) is

$$\begin{aligned} &\text{convex on } I_i, && \text{when } s_i < s_{i+1}, \\ &\text{concave on } I_i, && \text{when } s_i > s_{i+1}, \\ &\text{linear on } I_i, && \text{when } s_i = s_{i+1} = \delta_i, \end{aligned}$$

if and only if ξ_i is chosen in the interval I_i^C. Moreover, if I_i is an interval where the data are M-consistent, then choosing $\xi_i \in I_i^C$ guarantees that s is monotone on I_i.

Proof: The condition $\xi_i \in I_i^C$ is equivalent to the conditions given in Lemma 2.7 of [Schumaker '83]. ■

The above lemmas leave considerable freedom for choosing slopes $\{s_i\}_{i=1}^n$ such that the resulting spline s exhibits desirable shape properties (cf. the discussion in [Schumaker '83]). Two explicit methods have been proposed in the literature.

Lemma 3.5. *For $i = 2, ..., n-1$, let*

$$s_i = \begin{cases} 0, & \text{if } \delta_{i-1}\delta_i \le 0 \\ h_i, & \text{otherwise,} \end{cases} \tag{3.3}$$

where

$$h_i = \frac{2\delta_i\delta_{i-1}}{\delta_i + \delta_{i-1}}, \tag{3.4}$$

and δ_i is as in (3.2). For $i = 1$ and $i = n$, set

$$s_1 = \begin{cases} 0, & \text{if } \delta_1(2\delta_1 - s_2) \le 0 \\ 2\delta_1 - s_2, & \text{otherwise,} \end{cases} \tag{3.5}$$

$$s_n = \begin{cases} 0, & \text{if } \delta_{n-1}(2\delta_{n-1} - s_{n-1}) \le 0 \\ \\ 2\delta_{n-1} - s_{n-1}, & \text{otherwise.} \end{cases} \tag{3.6}$$

Let $\xi_i \in I_i^C$ for $i = 1, \ldots, n-1$. Then the spline s in (3.1) is co-monotone and co-convex in the following sense:

1) *if the data are M-consistent on I_i, then s is monotone on I_i;*

2) *if $\delta_\ell < \delta_{\ell+1} < \cdots < \delta_m$ and the data are C-consistent on $I_\ell, \ldots, I_{m-1}$, then s is convex on $[t_\ell, t_m]$;*

3) *if $\delta_\ell > \delta_{\ell+1} > \cdots > \delta_m$ and the data are C-consistent on $I_\ell, \ldots, I_{m-1}$, then s is concave on $[t_\ell, t_m]$.*

Proof: The first result of this type is due to [McAllister & Roulier '81]. In the generality stated here, see [Schumaker '83, DeVore & Yan '86]. ■

The quantity h_i appearing in (3.4) is the *harmonic mean* of δ_{i-1} and δ_i. It has also been used in [Butland '80] to assign slopes for a shape-preserving interpolation scheme based on C^1 cubic curves.

The McAllister & Roulier method described in Lemma 3.5 produces a spline which approximates a smooth function f with approximation order $\mathcal{O}(h^2)$, measured in the uniform norm on $[t_1, t_n]$, where h is the maximum spacing between the t_i's. In [DeVore & Yan '86], the authors presented two alternative methods with superior approximation properties. We describe their second method which provides order $\mathcal{O}(h^3)$ approximation on the entire interval $[t_1, t_n]$ (with a minor reduction in the shape-preserving properties).

Lemma 3.6. *For $i = 2, ..., n-1$, let*

$$s_i = \begin{cases} 0, & \text{if } \delta_i = 0 \text{ and } \delta_{i-1}\delta_{i+1} \ge 0 \\ 0, & \text{if } \delta_{i-1} = 0 \text{ and } \delta_{i-2}\delta_i \ge 0 \\ h_i, & \text{if } \delta_{i-1}\delta_i > 0 \text{ and } \min(d_i/\delta_i, d_{i+1}/\delta_i) \ge 2 \\ d_i, & \text{otherwise,} \end{cases} \tag{3.7}$$

where δ_i and h_i are defined as in Lemma 3.5, and where

$$d_i = (\delta_{i-1}\Delta t_i + \delta_i \Delta t_{i-1})/(\Delta t_{i-1} + \Delta t_i) \tag{3.8}$$

and

$$\Delta t_i = t_{i+1} - t_i. \tag{3.9}$$

For $i = 1$ and $i = n$, set

$$s_1 = 2\delta_1 - s_2, \tag{3.10}$$

$$s_n = 2\delta_{n-1} - s_{n-1}. \tag{3.11}$$

Let $\xi_i \in I_i^C$ for $i = 1, \ldots, n-1$. Then the spline s defined in (3.1) is co-monotone and co-convex in the following sense:

1) if the data are M-consistent on I_i, then s is monotone on I_i except

a) if $\delta_{i-1}\delta_i < 0$, then s may change direction on one of the two intervals I_{i-1} or I_i,

b) if $\delta_1 = 0$, s may change direction on I_1,

c) if $\delta_{n-1} = 0$, s may change direction on on I_{n-1},

2) if $\delta_\ell < \delta_{\ell+1} < \cdots < \delta_m$ and the data are C-consistent on $I_\ell, \ldots, I_{m-1}$, then s is convex on $[t_\ell, t_m]$,

3) if $\delta_\ell > \delta_{\ell+1} > \cdots > \delta_m$ and the data are C-consistent on $I_\ell, \ldots, I_{m-1}$, then s is concave on $[t_\ell, t_m]$.

Proof: The proof is a minor modification of the proof of Lemma 2 in [DeVore & Yan '86]. ■

The reason for the difference in shape properties of the methods of Lemma 3.5 and Lemma 3.6 is that the DeVore & Yan formula (3.7) assigns zero slopes less often. When s_{i-1} and s_{i+1} have opposite signs, we expect a change in the monotonicity of the spline to occur on the interval $[t_{i-1}, t_{i+1}]$. If s_{i-1} and s_{i+1} have opposite signs, the McAllister & Roulier formula (3.3) assigns zero slope at t_i, forcing the change in monotonicity of s on $[t_{i-1}, t_{i+1}]$ to occur at t_i. The DeVore & Yan formula does not make this zero slope assignment at t_i, and therefore does not unnecessarily force the change in monotonicity to occur at t_i. It follows that, as stated in $1a$), s may change direction on either I_{i-1} or I_i if $\delta_{i-1}\delta_i < 0$. If $\delta_1 = 0$ the McAllister & Roulier formula (3.5) forces the spline to be identically zero on the interval I_1 by assigning zero slopes at s_1 and s_2. The DeVore & Yan formula (3.10) does not place such a restriction on the slopes s_1 and s_2, and $1b$) follows. $1c$) follows similarly.

We conclude this section with an explicit algorithm for constructing an initial spline s satisfying $s(t_i) = z_i$, $i = 1, \ldots, n$ and preserving shape.

Algorithm 3.7.

For $i = 1, \ldots, n-1$

1) compute s_i using formula (3.7),

2) choose ξ_i to be the midpoint of the interval I_i^C,

3) compute the coefficients of $s|_{[t_i, t_{i+1}]}$ using the formulae in Lemma 2.1.

Discussion: This algorithm produces a C^1 quadratic spline with knots

$$\{\tau_1 < \tau_2 < \cdots < \tau_N\} = \{t_1 < \xi_1 < t_2 < \xi_2 < t_3 < \cdots < t_{n-1} < \xi_{n-1} < t_n\}.$$

If the data are C-consistent for I_i, then s is convex on I_i. If the data are not C-consistent for I_i, then $I_i^C = I_i^M$, and if the data are M-consistent for I_i, then s is monotone on I_i. If the data are neither M-consistent nor C-consistent for I_i, then $I_i^M = (t_i, t_{i+1})$ and ξ_i is chosen to be the midpoint of this interval. As mentioned in Lemma 3.4, $I_i^C \subseteq I_i^M$. Therefore, if the data are both M-consistent and C-consistent, choosing $\xi_i \in I_i^C$ guarantees that s is both monotone and convex on I_i. ■

§4. Preserving Shape in Algorithm 2.2

We are now in a position to suggest how to choose the new knot $\xi_j^{(k)}$ needed in step 1) of Algorithm 2.2:

1) if the interval $J_j^{(k-1)} = [\tau_j^{(k-1)}, \tau_{j+3}^{(k-1)}]$ does not contain any inflection points of the initial spline $s^{(0)}$ and the data at $\tau_j^{(k-1)}$ and $\tau_{j+3}^{(k-1)}$ are C-consistent, choose $\xi_j^{(k)}$ to be the midpoint of the associated convexity interval,

otherwise

2) choose $\xi_j^{(k)}$ to be the midpoint of the associated monotonicity interval.

In carrying out Algorithm 2.2, it can happen that after a number of steps, an interval $J_j^{(k-1)}$ does not contain any inflection points of the initial spline $s^{(0)}$, but the data at $\tau_j^{(k-1)}$ and $\tau_{j+3}^{(k-1)}$ are no longer C-consistent. The reason this can happen is that the lengths of the subintervals $J_j^{(k-1)}$ tend to increase with k, so that eventually the slope values at $\tau_j^{(k-1)}$ and $\tau_{j+3}^{(k-1)}$ may not be consistent with monotonicity and/or convexity on that interval. Thus, with the above choice of $\xi_j^{(k)}$, the spline $s^{(k)}$ may not be convex on $J_j^{(k-1)}$ even though the initial spline $s^{(0)}$ was. In practice this rarely happens.

We have tested Algorithm 2.2 using this knot selection scheme on a variety of problems, and have found that it has excellent shape-preserving properties. Some examples are presented in the following section.

If desired, we can exercise even more control over shape by altering the way in which weights are assigned in step 1) of Algorithm 2.2. For example, if $J_j^{(k-1)}$ does not contain any inflection points of the initial spline $s^{(0)}$, but the data at $\tau_j^{(k-1)}$ and $\tau_{j+3}^{(k-1)}$ are not C-consistent, then we could set $w_j^{(k)} = \infty$. This insures that if the initial spline was convex on $J_j^{(k-1)}$, then the new spline $s^{(k)}$ will also be. We have also tested this variant of the algorithm, and found that it produces fits which have about the same shape as the algorithm without the additional control, but it removes considerably fewer knots.

Since we are working with C^1 piecewise quadratic polynomials, it is easy to see that inflection points can only occur at knots. Thus, another way to control shape is to insist that at every step the inflection points remain in the same places. To do this, in examining the interval $J_j^{(k-1)}$, if either one of the points $\tau_{j+1}^{(k-1)}$ or $\tau_{j+2}^{(k-1)}$ was an inflection point of $s^{(0)}$, then we set $w_j^{(k)} = \infty$. We have also tested this variant of our algorithm, but it removes considerably fewer knots with only a minor improvement in shape. Indeed, the algorithm described above does an excellent job of keeping inflection points near their original locations, see Table 3.

We would like to emphasize that in Algorithm 2.2, the values of the slopes at all of the original knots remaining in the current spline approximation $s^{(k)}$ have not been changed from their original values. When a new knot is introduced, the slope associated with it is computed from formula (2.4). Now if in examining $J_j^{(k-1)}$, the slopes at its endpoints $\tau_j^{(k-1)}$ and $\tau_{j+3}^{(k-1)}$ are not consistent with the shape of $s^{(0)}$ on that interval, we could adjust the slopes at these endpoints. This would require constructing new spline pieces

on the intervals $[\tau_{j-2}^{(k-1)}, \tau_j^{(k-1)}]$ and $[\tau_{j+3}^{(k-1)}, \tau_{j+5}^{(k-1)}]$. We also tested this more complicated algorithm, but found that it does not perform significantly different from the algorithm described above.

§5. Numerical Examples

A FORTRAN implementation of Algorithm 2.2 was used to test the algorithm on several examples.

Example 5.1. *The function $f(x) = \sqrt{x}$ was sampled on a set D of 500 equally spaced points on the interval $[0, 1]$ to produce a set of data. The C^1 quadratic interpolating spline has 999 knots. Table 1 gives the number of interior knots remaining after applying Algorithm 2.2 for several values of the tolerance. It also shows the maximal error on the discrete set D.*

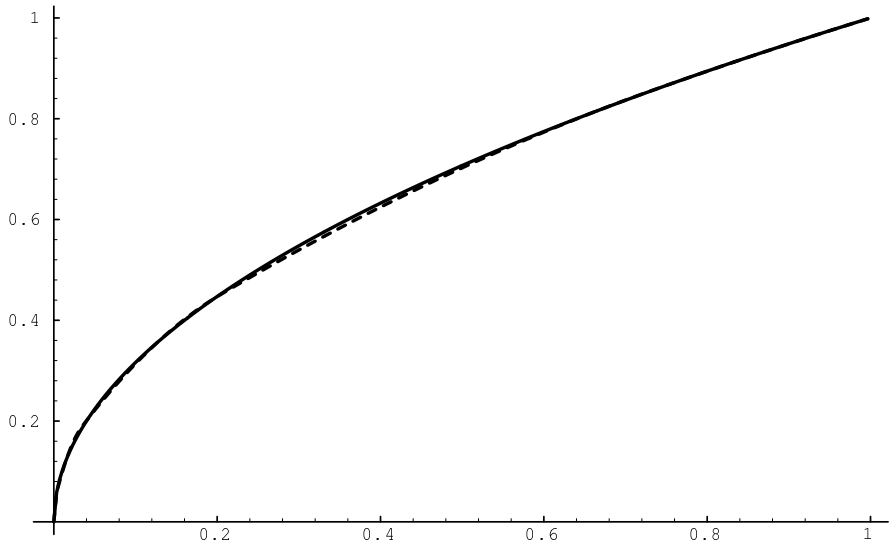

Figure 1. The splines with 999 knots (solid) and after removing 993 of them (dotted).

tolerance	error	# knots
0.0001	.0000540	23
0.001	.000985	10
0.01	.0087	4
0.1	.044	3

Table 1. Number of interior knots vs. tolerance for Example 5.1.

Fig. 1 shows the initial spline fit with 999 knots, and the spline corresponding to tolerance 0.01, where 993 knots have been removed. This example shows that even with a relatively small tolerance such as .0001, a large number of knots (976 out of 999) can be removed from the initial spline. Of course, we expect fewer knots to be needed as we increase the tolerance, but it is interesting to observe that for this example, each time the tolerance was increased by a factor of 10, approximately one-half of the remaining knots were removed.

Example 5.2. *The function* $f(x) = \frac{1}{x}\sin 5x$ *was sampled on a set* D *of 500 equally spaced points on the interval* $[0, 5]$ *to produce a set of data. The* C^1 *quadratic spline interpolating this data set has 999 knots. Table 2 gives the number of interior knots remaining after applying Algorithm 2.2 for several values of the tolerance. It also shows the maximal error on the discrete set* D.

tolerance	error	# knots
0.0001	.000097	134
0.001	.00093	67
0.01	.008	32
0.1	.084	14
0.5	.222	11

Table 2. Number of interior knots vs. tolerance for Example 5.2.

999 knots	13 knots
.4158408	0.4155353
1.187382	1.193565
1.838683	1.857244
2.479965	2.341548
3.111226	3.141577
3.752507	3.734161
4.383777	4.326746

Table 3. The inflection points of the splines in Figure 2.

The function in this example is considerably more complicated in shape than the one in Example 5.1. As before, each time we increased the tolerance by a factor of 10, we could remove approximately one-half of the remaining knots. Fig. 2 shows that even with a small number of remaining knots, our method does an excellent job of fitting the function while preserving the shape. For comparison, the inflection points for the splines in Figure 2 are given in Table 3.

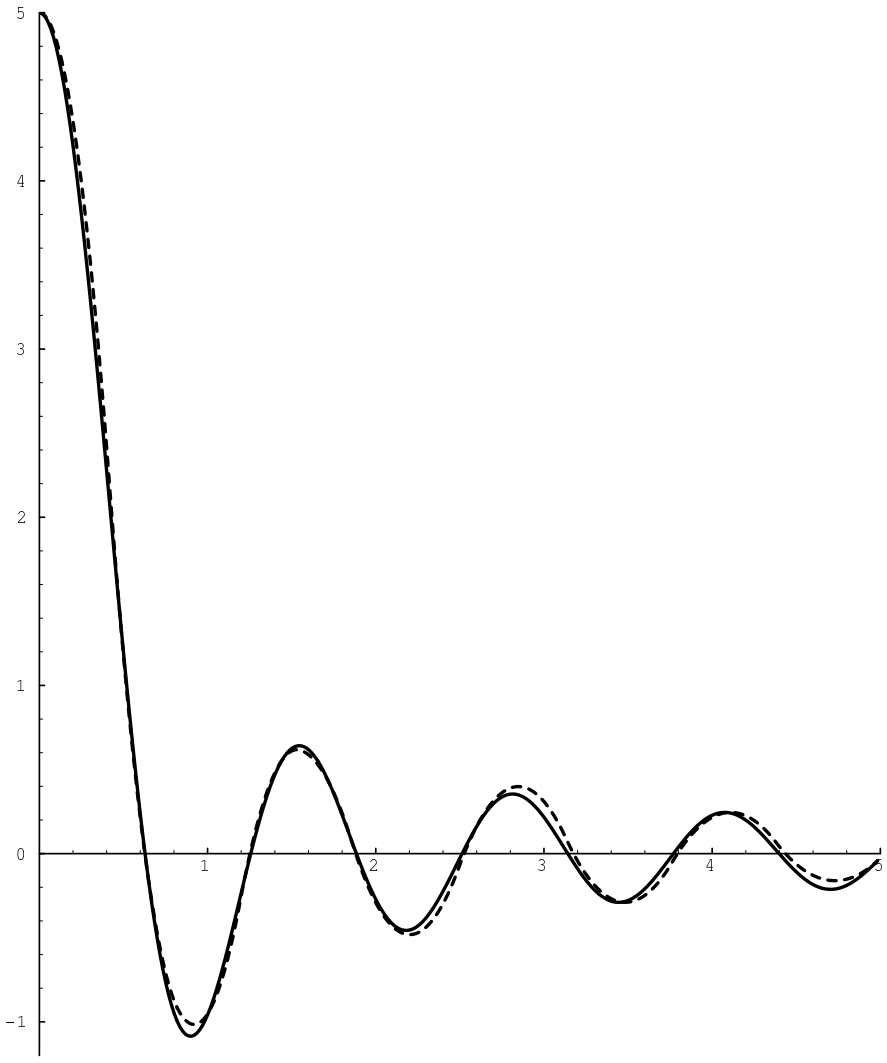

Figure 2. The splines with 999 knots (solid) and after removing 986 of them (dotted).

§6. Choosing Good Knots for Best Approximating Splines

One of the classical problems of spline approximation is the following: given a function f on an interval $[a, b]$ and an integer k, find a polynomial spline s of degree d with k knots in the interval $[a, b]$ such that s is a best approximation of f in some norm. We shall focus on the uniform norm (which in practice is measured on some discrete subset of $[a, b]$), and on the space of splines of degree 2.

This problem is called the *free knot* spline approximation problem. For theoretical results on it, see [Nürnberger '89] and references therein. Since this is a nonlinear approximation problem, it is not possible to find an exact solution computationally. In practice, we can proceed as follows:

1) choose some reasonable set of knots;

2) find the best approximation on that set (which can be done with a variant of the Remez algorithm, see [Nürnberger & Sommer '83, Nürnberger '89]);

3) adjust the knots and repeat.

In this section we explore the use of our shape-preserving knot removal algorithm as a way to find "good" knots. We proceed as follows:

1) sample f on some discrete subset of $[a, b]$. In practice we use 500 to 1000 equally spaced points;

2) construct a C^1 quadratic shape-preserving interpolant;

3) apply our knot removal algorithm with a series of larger and larger tolerances until the desired number of knots remain.

We now compare this approach with two other methods currently available in the literature:

1) *Segment approximation.* Here one approximates f by a space of piecewise polynomials (without any smoothness conditions) with k free knots. For theoretical results, and algorithms, see [Nürnberger & Sommer '83, Nürnberger '89] and [Wolters '93].

2) *deBoor's newnot algorithm* [de Boor '78].

To compare the three knot selection methods, we compute the best C^1 quadratic spline approximations to two typical functions. This is done for knot sets obtained by each of the three methods described above. For each example, we compute the maximum norms on 2000 equally spaced points in the domain of the function.

Our results are shown in Tables 4 and 5. In these tables, the columns labeled BSeg, BNN, and BSP give the errors of the best spline approximations based on the knots produced by segment approximation, *newnot*, and our shape-preserving method, respectively. The spline Remez algorithm of [Nürnberger & Sommer '83] (which was graciously provided to us by the authors) was used to compute best C^1 quadratic spline approximations in all three of these columns. For comparison purposes, in the column labeled Seg we give the error corresponding to the segment approximation itself (i.e., the approximation is a piecewise quadratic which may not even be continuous). We are grateful to H. Wolters for providing these numbers. In addition, in the column labeled SP, we give the error for our shape-preserving spline fit. It is, of course, always worse than the other splines, but as the examples show, it is often quite good.

Our first example deals with the function $f(x) = \sqrt{x}$ on $[0.00001, 1]$. Here we have chosen the left end point of the domain to be slightly larger than 0 because Wolters' segment approximation algorithm was not able to work on [0,1]. The results are shown in Table 4. The symbol NC means that our implementation of the spline Remez algorithm did not converge. In computing our shape-preserving spline, we began with an initial set of 2000 knots, where the first 1000 were equally spaced in the interval $[.00001, .1]$, and the remaining are equally spaced in the interval $[.1, 1]$. We used these initial knots in order to provide choices of knots which are close to zero.

For both segment approximation (Seg) and our shape-preserving method (SP), the error is monotone decreasing as the number of knots increases. We also note that the

# knots	Seg	BSeg	BNN	BSP	SP
1	.0164	.0193	.0319	.0422	.2913
2	.00644	.00867	.0153	.0198	.1138
3	.00313	.00448	.00969	.01244	.0329
4	.00174	.00256	.00706	.00661	.0286
5	.00108	NC	.00561	.00316	.00854
6	.000690	NC	.0047	.00207	.00472
7	.000473	.000716	.00406	.00173	.00234
8	.000366	.000513	.00376	.00173	.00218
9	.000249	NC	.00354	NC	.00202

Table 4. Comparison of knot selection methods for $f(x) = \sqrt{x}$.

errors for segment approximation (Seg) are always smaller than those of the best spline approximations based on the same knots (BSeg). Moreover, the errors for BSP are always smaller than those for SP, as they should be. As the table shows, BSP is better than BNN most of the time.

It is also worth noting that for this example, the error for our shape-preserving spline is less than 7 times as large as that of the best spline approximation based on the corresponding knots (the column SP is less than 7 times larger than BSP). This means that often we may be satisfied with our shape-preserving fit, and can save the work of calculating the best approximating spline.

Our second example deals with the classic Runge function $f(x) = 1/(1+x^2)$ on $[-5, 5]$. The results are shown in Table 5. As before, NC stands for no convergence in the spline Remez algorithm. Here NK means that we were not able to find a tolerance to produce the corresponding number of knots (by adjusting the tolerance we always got either one more or one less knot than desired). The shape-preserving splines were calculated starting with an initial set of 500 equally spaced knots.

For this example, the knots produced by segment approximation are generally superior, and those produced by *newnot* were second best, although the difference was not too large. For all three of the knot selection methods, it is possible for the error to increase as the number of knots is increased. Thus for example, in the column BSeg, the best spline approximation with 12 knots is better than that with 13 knots (remember, these are not the optimal knots, but are those produced by the segment approximation algorithm). Similarly, the error increases in column BNN in going from 14 to 15 knots.

§7. Fitting Noisy Data

In many applications, we want to fit an unknown function based on noisy measurements. In such cases, it does not make sense to construct an interpolating spline. The knot removal strategy of Algorithm 2.2 can still be applied in this situation, provided we have a good way to construct a C^1 quadratic spline to use as an initial fit. We can proceed as follows:

1) construct a preliminary spline fit g to the noisy data;

# knots	Seg	BSeg	BNN	BSP	SP
4	.00486	.0135	.0135	.0160	.0778
5	.00485	.0258	.0332	.0149	.0749
6	.00286	.00832	.0126	NK	NK
7	.00223	.00705	.00894	.0129	.0502
8	.00123	.00330	.00351	.00504	.0210
9	.00118	.00339	.00398	NC	.0198
10	.000693	NC	.00244	.00384	.0139
11	.000557	.00156	.00222	.00253	.0137
12	.000463	.00123	.00115	.00249	.00853
13	.000433	.00135	.00161	.00192	.00802
14	.000251	NC	.000786	.00190	.00695
15	.000240	.000671	.000902	.00190	.00401
16	.000213	.000567	NC	.00127	.00361
17	.000197	.000532	.000622	NC	.00205
18	.000136	NC	.000306	.00127	.00174
19	.000135	NC	.000449	.000795	.00117

Table 5. A comparison of knot selection methods for $f(x) = 1/(1+x^2)$.

2) sample this spline on a fine mesh of equally spaced points and construct the corresponding shape-preserving C^1 quadratic interpolant s;

3) remove knots from s.

We suggest using penalized least squares for step 1). Suppose $y_i = f(x_i) + \epsilon_i$, $i = 1, \ldots, n$, are measurements on some unknown function f with measurement errors $\{\epsilon_i\}$, $i = 1, \ldots, n$. To compute an approximation to f, we seek g in the form

$$g_{\mathbf{c}}(x) = \sum_{i=1}^{k} c_i B_i(x), \tag{7.1}$$

where B_i are the usual cubic B-splines associated with some knot vector (usually equally spaced). To find an acceptable approximation, we must choose a suitable vector of coefficients.

The classical least squares method involves the minimization of

$$E(\mathbf{c}) = \frac{1}{n} \sum_{i=1}^{n} [g_{\mathbf{c}}(x_i) - y_i]^2$$

over all choices of coefficient vectors $\mathbf{c}$. For some data fitting problems, this results in fitting functions that are not sufficiently smooth. The idea of *penalized least squares* [Golitschek & Schumaker '90] is to construct a good fit to the data by choosing the coefficient vector to minimize a combination of measures of goodness of fit and smoothness. We can take

the usual ℓ_2 norm for the measure of goodness of fit. The smoothness of a function g on an interval $[a, b]$ can be measured by

$$J(g) = \int_a^b [D^m g(t)]^2 dt.$$

Now given $\lambda > 0$, we seek to minimize

$$\rho_\lambda(\mathbf{c}) = \lambda J(g) + \frac{1}{n}\sum_{i=1}^{n}[g(x_i) - y_i]^2$$

over our spline space. The first term of $\rho_\lambda(\mathbf{c})$ is called the *penalty term*, and λ is called the *smoothing parameter*.

For splines $g_\mathbf{c}$ expressed in the form (7.1) the energy term can be written as $J(g_\mathbf{c}) = \mathbf{c}^T E\mathbf{c}$, where

$$E_{ij} = \int_a^b D^m B_i(t) D^m B_j(t) dt, \quad i, j = 1, ..., k.$$

The efficient computation of E is discussed in [Golitschek & Schumaker '90]. Now our objective function has the form

$$\rho_\lambda(\mathbf{c}) = \lambda \mathbf{c}^T E\mathbf{c} + \frac{1}{n}\sum_{i=1}^{n}[g_\mathbf{c}(x_i) - y_i]^2.$$

The following result can be found in [Golitschek & Schumaker '90].

Theorem 7.1. *Let the $n \times k$ observation matrix B with entries $B_{ij} = B_j(x_i)$, $i = 1, ..., n$, $j = 1, ..., k$, be given and assume the matrix B is such that*

$$det[B_j(x_{i_v})]_{v=1,j=1}^{k,k} \neq 0, \quad where \quad \{x_{i_v}\}_{v=1}^{k} \subset \{x_i\}_{i=1}^{n}.$$

For any $\lambda \geq 0$ there exists a unique vector $\mathbf{c}(\lambda)$ minimizing $\rho_\lambda(\mathbf{c})$. It is the unique solution of the system

$$(B^T B + n\lambda E)\mathbf{c}(\lambda) = B^T \mathbf{y}, \tag{7.2}$$

where $\mathbf{y} = (y_1, ..., y_n)^T$.

In practice, one has to select the smoothing parameter λ. The standard approach is to use generalized cross validation (see the discussion in [Golitschek & Schumaker '90]). For the use of Monte-Carlo methods for finding λ, see [Girard '89].

Figure 3 illustrates the use of penalized least squares to fit noisy data. First we created a set of noisy data by adding random numbers in the interval [-.7,.7] to the values of the function $f(x) = \frac{1}{x}\sin 5x$ at 200 equally spaced points in [0,5]. Next we computed a cubic penalized least squares spline fit ($m = 2$) to the noisy data using 9 equally spaced interior knots and $\lambda = 0.001$. We sampled this cubic spline at 200 equally spaced points to create a quadratic C^1 shape-preserving spline, and then applied our knot removal algorithm with tolerance .05. The resulting spline (which has 19 interior knots) is shown in Fig. 3.

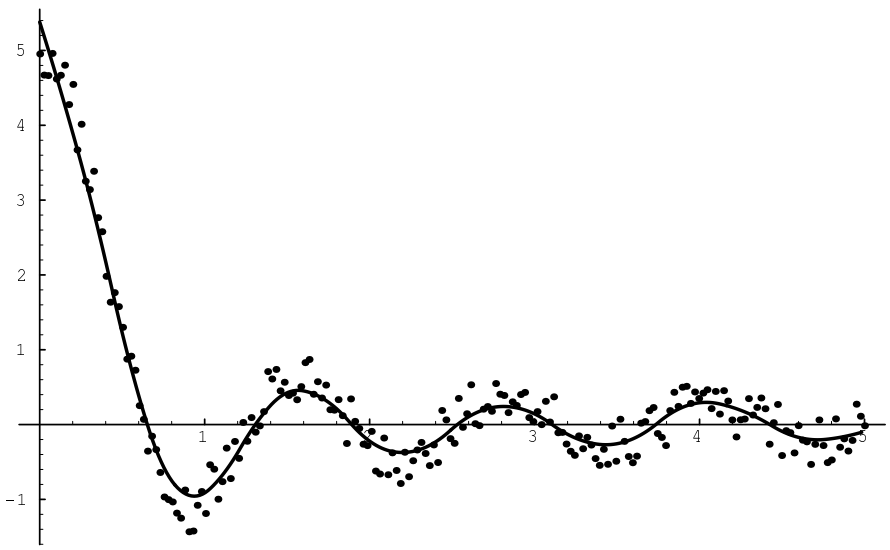

Figure 3. Noisy data and its corresponding spline fit.

§8. Knot Removal for Bivariate Functions

In this and the following section, we discuss monotonicity preserving knot removal for bivariate functions. First we need to define what we mean by monotone data.

Definition 8.1. *Let $D = \{(x_i, y_i, z_i)\}_{i=1}^N \subset \mathbb{R}^3$ be a finite data set. D is a monotone increasing data set provided that $z_j \geq z_i$ for all points (x_i, y_i) and (x_j, y_j) such that $x_j \geq x_i$ and $y_j \geq y_i$.*

A monotone decreasing data set is defined similarly. In this paper we will direct our attention to the case of monotone increasing data sets since the monotone decreasing case is similar.

Our method is based on an interpolation method of [Han & Schumaker '95] which produces a monotone surface given monotone increasing data on a grid. Suppose H is a rectangular grid defined by $\{x_i\}_{i=1}^{nx}$, $\{y_j\}_{j=1}^{ny}$, and suppose $\{z_{ij}\}_{i=1,j=1}^{nx,ny}$ are corresponding real numbers. We suppose this data is monotone in the sense of Definition 8.1. Now suppose we are given

$$\{z_{ij}^x\}_{i=1,j=1}^{nx,ny}, \quad \{z_{ij}^y\}_{i=1,j=1}^{nx,ny}$$

at each of the grid points, and let Δ be the triangulation obtained from the grid by drawing in both diagonals in each subrectangle $H_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ of H. This subdivision of H_{ij} is called the *Sibson split* of H_{ij}.

It was shown in [Han & Schumaker '95] that there exists a unique spline s in the space $\mathcal{S}$ of cubic C^1 splines on Δ whose normal derivatives along the edges are linear instead of

quadratic which satisfies

$$s(x_i, y_j) = z_{ij}, \quad s_x(x_i, y_j) = z^x_{ij}, \quad s_y(x_i, y_j) = z^y_{ij} \tag{8.1}$$

for $i = 1, ..., nx$ and $j = 1, ..., ny$.

Now, as in the univariate case, even though the data set is monotone, the above interpolating spline will not be monotone for arbitrary choices of the gradients. The following theorem from [Han & Schumaker '95] gives sufficient conditions on the gradients for s to be monotone in the sense that

$$s(\tilde{x}, \tilde{y}) \geq s(x, y),$$

whenever $\tilde{x} \geq x$, $\tilde{y} \geq y$.

Theorem 8.2. *Suppose we are given data as in (8.1) such that*

$$z^x_{ij} + z^x_{i+1,j} \leq 5(z_{i+1,j} - z_{ij})/2h^x_i, \qquad z^x_{i,j+1} + z^x_{i+1,j+1} \leq 5(z_{i+1,j+1} - z_{i,j+1})/2h^x_i,$$

$$z^x_{ij} \leq z^x_{i,j+1} + \min\left\{ \frac{3}{2h^x_i}(z_{i,j+1} - z_{ij}), \frac{6}{h^x_i}(z_{i,j+1} - z_{ij}) - \frac{2h^y_j}{h^x_i} \max[z^y_{ij}, z^y_{i,j+1}] \right\},$$

$$\begin{aligned} z^x_{i+1,j+1} &\leq z^x_{i+1,j} \\ &+ \min\left\{ \frac{3}{2h^x_i}(z_{i+1,j+1} - z_{i+1,j}), \frac{6}{h^x_i}(z_{i+1,j+1} - z_{i+1,j}) - \frac{2h^y_j}{h^x_i} \max[z^y_{i+1,j}, z^y_{i+1,j+1}] \right\}, \end{aligned}$$

where $h^x_i = (x_{i+1} - x_i)$ and $h^y_i = (y_{j+1} - y_j)$. Suppose that similar conditions hold for the y partial derivatives. Then the spline s_{ij} which interpolates as in (8.1) is monotone on H_{ij}.

Based on this theorem, the following two-step procedure is presented in [Han & Schumaker '95] for choosing the gradients $\{z^x_{ij}\}^{nx,ny}_{i=1,j=1}$ and $\{z^y_{ij}\}^{nx,ny}_{i=1,j=1}$ so that the interpolating surface s is monotone increasing:

1) choose some initial set of nonnegative $\{z^x_{ij}\}^{nx,ny}_{i=1,j=1}$ and $\{z^y_{ij}\}^{nx,ny}_{i=1,j=1}$ using quadrature rules based on local polynomials;
2) adjust these values so that the eight inequalities of Theorem 8.2 are satisfied for each subrectangle.

Step 2 involves several sweeps of the grid to force the conditions of Theorem 8.2 to be satisfied.

Once we have a monotone surface which interpolates the gridded data points, we remove knot lines from the underlying grid in such a way that the surface is not perturbed more than the given tolerance, and so that the monotonicity of the surface is also preserved. This proceeds as follows:

Algorithm 8.3. Let $s^{(0)}$ be a spline constructed as above which interpolates the data $\{z_{ij}\}_{i=1,j=1}^{nx,ny}$, $\{z_{ij}^x\}_{i=1,j=1}^{nx,ny}$, and $\{z_{ij}^y\}_{i=1,j=1}^{nx,ny}$, at $\{(x_i,y_j)\}_{i=1,j=1}^{nx,ny}$. Let $tol > 0$:

1) *let* $\mathbf{x^0} = \{x_i\}_{i=1}^{nx}$, $\mathbf{y^0} = \{y_j\}_{j=1}^{ny}$, $\mathbf{z^0} = \{z_{ij}\}_{i=1,j=1}^{nx,ny}$, $\mathbf{zx^0} = \{z_{ij}^x\}_{i=1,j=1}^{nx,ny}$, *and* $\mathbf{zy^0} = \{z_{ij}^y\}_{i=1,j=1}^{nx,ny}$;

2) *form* $\mathbf{wx^0}$ *and* $\mathbf{wy^0}$, *vectors of weights (see the discussion) of the knots in* $\mathbf{x^0}$ *and* $\mathbf{y^0}$, *respectively;*

3) $i = 0$;

4) *do while* $|\mathbf{x^i}| \geq 4$ *and* $|\mathbf{y^i}| \geq 4$, *with either* $|\mathbf{x^i}| > 4$, $|\mathbf{y^i}| > 4$, *or both, and* $min\{\mathbf{wx^i} \cup \mathbf{wy^i}\} < tol$

   a) *form* $\mathbf{x^{i+1}} = \mathbf{x^i} - \bar{\mathbf{x}}^\mathbf{i}$ *and* $\mathbf{y^{i+1}} = \mathbf{y^i} - \bar{\mathbf{y}}^\mathbf{i}$, *where* $\bar{\mathbf{x}}^\mathbf{i} \subset \mathbf{x^i}$ *and* $\bar{\mathbf{y}}_\mathbf{i} \subset \mathbf{y^i}$ *are chosen according to the strategy given in the discussion below,*

   b) *form* $\mathbf{z^{i+1}}$, $\mathbf{zx^{i+1}}$, *and* $\mathbf{zy^{i+1}}$ *by properly adjusting* $\mathbf{z^i}$, $\mathbf{zx^i}$, *and* $\mathbf{zy^i}$,

   c) *construct the surface* $s^{(i+1)}$,

   d) *find* $\mathbf{wx^{i+1}}$ *and* $\mathbf{wy^{i+1}}$,

   e) $i = i + 1$.

Discussion: Each interior knot of $\{x_i\}_{i=1}^{nx}$ and $\{y_j\}_{j=1}^{ny}$ is weighted to reflect its importance, and in doing this we create two vectors of weights $\mathbf{wx^0}$ and $\mathbf{wy^0}$. The weights in $\mathbf{wx^0}$ are calculated by temporarily removing one interior knot x_k from $\{x_i\}_{i=1}^{nx}$, building the surface $sx_k^{(0)}$ without the knot line created by x_k, and setting $wx_k^0 = \|s^{(0)} - sx_k^{(0)}\|$. We use the discrete uniform norm calculated by finding the maximum value of $|s^{(0)} - sx_k^{(0)}|$ on a fine mesh of grid points in H. We do the same for each interior knot of $\{y_j\}_{j=1}^{ny}$ to form the vector of weights $\mathbf{wy^0}$. This process is also used to form $\mathbf{wx^i}$ and $\mathbf{wy^i}$.

Calculating the weight of each knot is computationally more expensive in the bivariate case than in the univariate case because each time a knot line is removed from the underlying grid, the monotone surface interpolating the remaining gridded data points must be constructed. Therefore, to reduce the number of computations, we suggest removing several knots at a time. While $|\mathbf{x^i}| \geq 4$ and $|\mathbf{y^i}| \geq 4$, with either $|\mathbf{x^i}| > 4$, $|\mathbf{y^i}| > 4$, or both, and $min\{\mathbf{wx^i} \cup \mathbf{wy^i}\} < tol$, we first try to remove every other knot from $\mathbf{x^i}$ with weight less than $tol/2$, and every other knot from $\mathbf{y^i}$ with weight less than $tol/2$. This insures that no two adjacent knots will be removed. We construct the interpolating surface with the remaining information and if the resulting error is less than the tolerance we go back to the beginning of step 4) and try to remove more knots. If the error is greater than or equal to the tolerance, we try removing every fourth knot, every eighth knot, and so on, with weight less than $tol/2$. When there are fewer than three knots in either sequence with weights less than $tol/2$, we use tol instead of $tol/2$ in the knot removal decision. We can usually remove a few more knots using this criterion at this point in the algorithm. Experiments show that we need to be selective about which knots are removed in the beginning of the knot removal process in order for us to be able to remove as many knots as possible.

When the surface is reconstructed following the removal of one or more knot lines, the values for the gradients are not recomputed before the sweeps of the grid are administered.

As in the univariate case, we prefer to use the original position and gradient values as often as possible. ■

§9. Numerical Examples for Surfaces

A FORTRAN implementation of Algorithm 8.3 was used to test the algorithm on several examples.

Example 9.1. The function

$$f(x,y) = \begin{cases} exp(\frac{-1.0}{(\sqrt{x^2+y^2}-0.6)^2}), & \text{if } \sqrt{x^2+y^2} > 0.6 \\ 0, & \text{otherwise,} \end{cases}$$

was sampled at 31×31 equally spaced grid points on the rectangle $[0,3] \times [0,3]$ to produce a set of data. Table 6 gives the number of knots remaining after applying Algorithm 8.3 for several values of the tolerance.

tolerance	# knots in **x**	# knots in **y**
0.0001	31	31
0.001	25	25
0.01	11	11
0.1	4	4

Table 6. Number of knots vs. tolerance for Example 9.1.

Example 9.2. The function $f(x,y) = x^4 + y^2$ was sampled at 31×31 equally spaced grid points on the rectangle $[0,3] \times [0,3]$ to produce a set of data. Table 6 gives the number of knots remaining after applying Algorithm 8.3 for several values of the tolerance.

tolerance	# knots in **x**	# knots in **y**
0.0001	29	15
0.001	17	17
0.01	11	8
0.1	8	8

Table 7. Number of knots vs. tolerance for Example 9.2.

10. Remarks

Remark 1. Algorithm 2.2 removes one knot at a time, but it is also possible to remove a group of knots at a time. When removing knots in groups, we suggest an approach similar to that in Algorithm 8.3, which is to insure that a group of knots to be removed contain no adjacent knots.

Remark 2. The order in which the knots are removed from an initial spline affects both the number and locations of the final knots. To get a fit with the smallest possible number of knots, it may be necessary to sometimes remove knots which do not have the smallest weights. To find the best path would then require conducting some kind of combinatorial search. Simulated annealing could possibly be applied.

Remark 3. Our bivariate shape-preserving knot removal method can also be applied to noisy data. Just as in the univariate case, this can be done by first approximating the noisy data with a tensor-product B-spline surface (for example, by penalized least squares), and then sampling that surface to create values to be used to construct an initial interpolating surface. For details on how to do tensor-product penalized least squares, see [Weyrich '92].

Remark 4. Our bivariate knot removal algorithm can also be regarded as a way to select good knots for a tensor-product spline approximation of a bivariate function. In this case, however, we have to be satisfied with the shape-preserving approximation itself, since there is no known algorithm for computing best tensor-product spline approximations (in the uniform norm).

References

Arge, E., M. Daehlen, T. Lyche, and K. Mørken (1990), Constrained spline approximation of functions and data based on constrained knot removal, in *Algorithms for Approximation II*, M. G. Cox and J. C. Mason (eds.), Chapman & Hall (London), 4–20.

deBoor, C. (1978), *A Practical Guide to Splines*, Springer-Verlag, New York.

Butland, J. (1980), A method of interpolating reasonable-shaped curves through any data, Proc. Computer Graphics 80, Online Publications Ltd., Middlesex, UK, 409–422.

DeVore, R., and Z. Yan (1986), Error analysis for piecewise quadratic curve fitting algorithms, Comput. Aided Geom. Design **3**, 205–215.

Girard, D. (1989), A fast 'Monte-Carlo cross-validation' procedure for large least squares problems with noisy data, Numer. Math. **56**, 1–23.

Goldman, R., and T. Lyche, eds. (1993), *Knot Insertion and Deletion Algorithms for B-spline Curves and Surfaces*, SIAM (Philadelphia).

von Golitschek, M., and L. Schumaker (1990), Data fitting by penalized least squares, in *Algorithms for Approximation II*, M. G. Cox and J. C. Mason (eds.), Chapman & Hall (London), 210–227.

Han, L., and L. Schumaker (1995), Fitting monotone surfaces to scattered data using C^1 piecewise cubics, SIAM J. Numer. Anal., to appear.

Lyche, T. (1992), Knot removal for spline curves and surfaces, in *Approximation Theory VII*, E. W. Cheney, C. Chui, and L. Schumaker (eds.), Academic Press (New York), 1–21.

Lyche, T., and K. Mørken (1987a), Knot removal for parametric B-spline curves and surfaces, Comput. Aided Geom. Design **4**, 217–230.

Lyche, T., and K. Mørken (1987b), A discrete approach to knot removal and degree reduction algorithms for splines, in *Algorithms for Approximation II*, M. G. Cox and J. C. Mason (eds.), Chapman & Hall (London), 67–82.

Lyche, T., and K. Mørken (1988), A data reduction strategy for splines, IMA J. Numer. Anal. **8**, 185–208.

McAllister, D. F., and J. A. Roulier (1981), An algorithm for computing a shape-preserving osculatory quadratic spline, ACM Trans. Math. Software **7**, 331–347.

Nürnberger, G. (1989), *Approximation by Spline Functions*, Springer-Verlag, Berlin.

Nürnberger, G., and M. Sommer (1983), A Remez Type Algorithm for Spline Functions, Numer. Math. **41**, 117–146.

Nürnberger, G., M. Sommer and H. Strauss (1986), An algorithm for segment approximation, Numer. Math. **48**, 463–477.

Schumaker, L. (1983), On shape-preserving quadratic spline interpolation, SIAM J. Numer. Anal. **20**, 854–864.

Weyrich, N. (1992), Bivariate spline approximation by penalized least squares, in *Mathematical Methods in Computer Aided Geometric Design II*, T. Lyche and L. Schumaker (eds.), Academic Press (New York), 607–614.

Wolters, H. (1993), A method for computing best segment approximations, Arizona State University, TR 93-001.