Piecewise Cubic Curve Fitting Algorithm

By Zheng Yan

Abstract. We give a piecewise cubic curve fitting algorithm which preserves monotonicity of the data. The algorithm has a higher order of convergence than the Fritsch-Carlson algorithm and is simpler than the Eisenstat-Jackson-Lewis algorithm.

1. Introduction. We are interested in numerically fitting a curve through a given finite set of points $P_i = (x_i, y_i)$, $i = 0, \ldots, n$, in the plane, with $0 = x_0 < x_1 < \cdots < x_n = 1$. These points can be thought of as coming from the graph of some function f defined on $[0, 1]$. We are particularly interested in algorithms which preserve local monotonicity of the data (or function), i.e., if $\Delta y_i := y_{i+1} - y_i \geq 0$, $a \leq i < b$, then we want the resulting interpolant to be monotone on (x_a, x_b).

The best known algorithms of this type [5], [4], generate a piecewise polynomial S (quadratic in [5] and cubic in [4]) such that $y = S(x)$ is the desired interpolant. Here, we shall focus only on those algorithms which use C^1 piecewise cubics S. The Fritsch-Carlson algorithm [4] and the algorithms discussed in Eisenstat, Jackson and Lewis [3] are of this type. These two algorithms generate a C^1 piecewise polynomial whose knots are at the points x_i, $i = 1, \ldots, n - 1$.

It is well known that C^1 piecewise cubics can approximate a four times continuously differentiable f to an order $O(h^4)$ with h the maximum spacing of the knots. We are interested here whether the curve fitting algorithms of the type described above can also have this order of approximation. It turns out (as was noted in [3]) that the FC algorithm is of order $O(h^3)$. On the other hand, the EJL algorithm has the highest possible order of convergence but it is rather complicated and, in particular, it requires two sweeps through the data to generate the interpolant.

The purpose of the present paper is to propose what we feel is an attractive alternative to these algorithms. Its main advantage over the EJL algorithm is that it is much simpler to implement, since it requires only a single sweep through the data and yet has the same monotonicity-preserving property and the same order of convergence.

The typical algorithm for generating a C^1 cubic interpolant which preserves monotonicity has the following form. It begins with some assignment of slopes s_i, at the points x_i, $i = 0, \ldots, n$, which is third-order accurate when the underlying function f is smooth (four times continuously differentiable). After this assignment,

the Hermite cubic H which interpolates the values y_i and derivative values s_i, $i = 0, \ldots, n$, is taken as the first try for the interpolant. It may happen that H does not agree in monotonicity with the data. This typically happens when the data changes slowly, i.e., where f' is small. In the FC and EJL algorithms this is overcome by changing the slope assignment s_i where this occurs. Our approach is different in that, rather than changing the slopes, we insert additional knots ξ_{i1}, ξ_{i2} in each interval (x_i, x_{i+1}) where the Hermite interpolant does not agree in monotonicity with the data. We then modify the interpolant so that it is a C^1 cubic with respect to the expanded set of knots. Part of the simplicity of our algorithm comes from making our interpolant constant on the intervals (ξ_{i1}, ξ_{i2}). This means that the graph of S will possibly be flat on intervals where the underlying function has a small derivative.

2. Piecewise Cubics and Monotonicity in a Single Interval for Cubic Polynomials.
The knots of a piecewise polynomial S are the points of discontinuity of S or its derivatives. We are interested in piecewise cubics S which are in $C^1[0,1]$, which interpolate our data (x_i, y_i), $i = 0, \ldots, n$, and which have knots at the points x_i, $1 \leqslant i < n$. Such piecewise cubics are completely determined by their values $y_i = S(x_i)$ and their derivative values $s_i := S'(x_i)$, $i = 0, 1, \ldots, n$. Since S is a cubic polynomial on each interval $[x_i, x_{i+1}]$, which satisfies the interpolation conditions, the derivatives $s_i = S'(x_i)$ and $s_{i+1} = S'(x_{i+1})$ completely determine S on $[x_i, x_{i+1}]$. In fact, we have

LEMMA 2.1. *Let* π: $0 = x_0 < x_1 < \cdots < x_n = 1$ *be a partition of the interval* $I = [0, 1]$. *Let* $\{y_i\}$ *be a given set of data. Set* $\Delta x_i := x_{i+1} - x_i$ *and* $\delta_i := (y_{i+1} - y_i)/\Delta x_i$. *Then for each interval* $[x_i, x_{i+1}]$, S *can be represented as*

$$\text{(i)} \quad S(x) = \frac{[s_i + s_{i+1} - 2\delta_i]}{(\Delta x_i)^2}(x - x_i)^3$$
$$+ \frac{[-2s_i - s_{i+1} + 3\delta_i]}{\Delta x_i}(x - x_i)^2 + s_i(x - x_i) + y_i$$

and

$$\text{(ii)} \quad S'(x) = \frac{[3(s_i + s_{i+1} - 2\delta_i)]}{(\Delta x_i)^2}(x - x_i)^2$$
(2.1)
$$+ \frac{[2(-2s_i - s_{i+1} + 3\delta_i)]}{\Delta x_i}(x - x_i) + s_i$$

and

$$\text{(iii)} \quad S''(x) = \frac{[6(s_i + s_{i+1} - 2\delta_i)]}{(\Delta x_i)^2}(x - x_i)$$
$$+ \frac{[2(-2s_i - s_{i+1} + 3\delta_i)]}{\Delta x_i}.$$

Proof. This is the Newton representation for an interpolation polynomial. □

We are interested in characterizing when S is monotone in terms of the numbers s_i, s_{i+1} and δ_i. For this, we follow the analysis of Fritsch and Carlson [4]. It is clear that a necessary condition for monotonicity of S on $I_i = [x_i, x_{i+1}]$ is that

(2.2) $$\operatorname{sgn}(s_i) = \operatorname{sgn}(s_{i+1}) = \operatorname{sgn}(\delta_i).$$

Further, if $\delta_i = 0$, then S is monotone (i.e., constant) on I_i if and only if $s_i = s_{i+1} = 0$. For the remainder of this section we shall therefore assume that $\delta_i \neq 0$.

It turns out that a characterization of when S is monotone depends on the sign of the number $\alpha_i + \beta_i - 2$, where $\alpha_i := s_i/\delta_i$, $\beta_i := s_{i+1}/\delta_i$. Then we have the following lemmas [4, Lemmas 1 and 2]:

LEMMA 2.2. *If $\alpha_i + \beta_i - 2 \leqslant 0$, then S is monotone on I_i if and only if (2.2) is satisfied.*

LEMMA 2.3. *If $\alpha_i + \beta_i - 2 > 0$, then S is monotone on I_i if and only if (2.2) is satisfied and one of the following conditions is satisfied:*
   (i) $2\alpha_i + \beta_i - 3 \leqslant 0$;
   (ii) $\alpha_i + 2\beta_i - 3 \leqslant 0$; *or*
   (iii) $\phi(\alpha_i, \beta_i) \geqslant 0$,
where $\phi(\alpha, \beta) = \alpha - (2\alpha + \beta - 3)^2/3(\alpha + \beta - 2)$.

As a consequence of Lemmas 2.2 and 2.3, it is possible to construct a region M of acceptable values for α_i and β_i (hence s_i and s_{i+1}) which produce a monotone interpolant on I_i. This region is shown in Figure 2-1 along with associated exterior regions A, B, C, D and E. All regions are closed. We note that the curve $\phi(\alpha, \beta) = 0$ is the ellipse $(\alpha - 1)^2 + (\alpha - 1)(\beta - 1) + (\beta - 1)^2 - 3(\alpha + \beta - 2) = 0$, which is tangent to the coordinate axes at $(3, 0)$ and $(0, 3)$ [4].
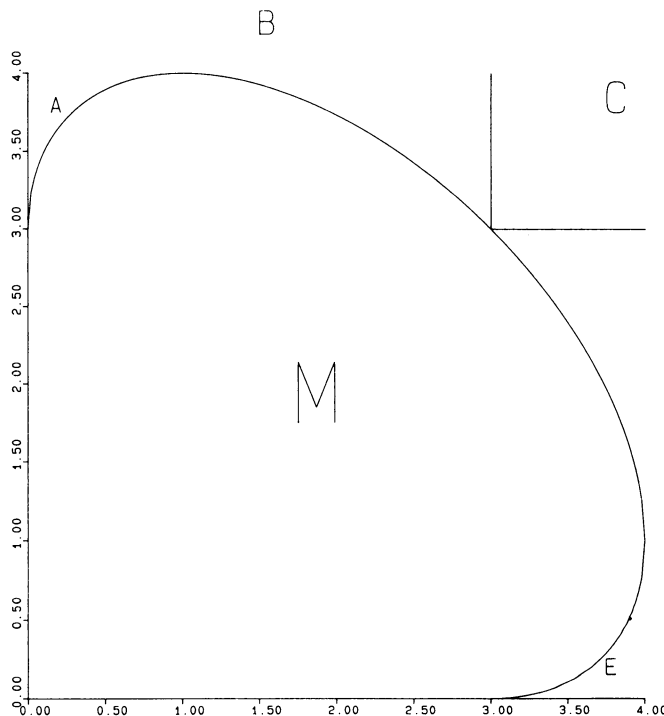

FIGURE 2-1

The above Lemmas 2.2 and 2.3 mean that S is monotone on I_i if and only if $(s_i, s_{i+1}) \in M_i$, where $M_i = M\delta_i = \{(x\delta_i, y\delta_i): (x, y) \in M\}$.

3. Existing algorithms for monotonicity-preserving cubic interpolation.

The existing algorithms for piecewise cubic monotonicity-preserving interpolation consist of assigning values s_i for the slopes of S at x_i and then modifying these to guarantee monotonicity. Fritsch and Carlson give the FC algorithm which assumes that some reasonable initial assignment has been made for the slopes s_i, $i = 0, 1, \ldots, n$, and some region $W \subset M$ has been chosen with the property that if $(x, y) \in W$ and $(u, v) \leqslant (x, y)$ (that is, $u \leqslant x$ and $v \leqslant y$) then (u, v) is also in W. A standard choice of W is a square which is bounded by the four lines $\alpha = 0, 3$ and $\beta = 0, 3$. It was noted by Eisenstat, Jackson and Lewis [3] that for the FC algorithm to be second-order accurate, the point $(1, 1)$ must be in W; for the algorithm to be third-order accurate, the closed triangle with vertices $(0, 0)$, $(2, 0)$, $(0, 2)$ must be contained in W.

Eisenstat, Jackson and Lewis have given two modifications of FC which improve the order of convergence. The first is the *Two-Sweep Algorithm*. In their algorithm they modify $\{s_i\}$ so that each ordered pair $(s_i, s_{i+1}) \in M_i$. This is accomplished by making two sweeps through the data. The first, a forward sweep, modifies only the second component. Then this is followed by a backward sweep which modifies the first component only and then guarantees that (s_i, s_{i+1}) is in M_i. The two-sweep algorithm results in third-order convergence (that is $O(h^3)$) when the data are chosen from the graph of a smooth function. The second is the *Extended Two-Sweep Algorithm*. This algorithm is even more complicated but results in fourth-order convergence.

In the following section we describe what we consider to be an attractive alternative to these three algorithms. In contrast to the above algorithms, we will not modify the values of the s_i. Instead, when (s_i, s_{i+1}) is not in M_i, we shall insert two knots in the interval (x_i, x_{i+1}). In this simpler way, we derive a one-pass algorithm which has fourth-order convergence.

4. A New Algorithm for Monotonicity-Preserving Cubic Interpolation.

In this new algorithm, we base our curve fitting on the idea of inserting new knots. Let s_i be some initial assignment of slopes at the points x_i, $i = 0, \ldots, n$, such that s_i is compatible with δ_{i-1} and δ_i, that is s_i, s_{i+1} and δ_i are not of opposite signs. Let S_0 be the C^1 cubic spline with knots $\{x_i\}$ which interpolates our data, that is, S_0 satisfies $S_0(x_i) = y_i$ and $S_0'(x_i) = s_i$, $i = 0, \ldots, n$. When S_0 is not monotone on an interval $I_i := [x_i, x_{i+1}]$, that is when $(\alpha_i, \beta_i) \notin M_i$, then we shall modify S_0 on I_i by inserting two additional knots ξ_{i1}, $\xi_{i2} \in (x_i, x_{i+1})$. The following discussion will describe how these knots are chosen. Now S_0, S_0', and S_0'' can be represented as in (2.1) on each interval I_i. We observe that the quadratic polynomial Q of (2.1)(ii) which represents S_0' on (x_i, x_{i+1}) has a unique extremum at

$$(4.1) \qquad x^* = x_i + \Delta x_i (2\alpha_i + \beta_i - 3)/3(\alpha_i + \beta_i - 2).$$

Since $S_0'(x_i)S_0'(x_{i+1}) \geqslant 0$ and yet $(\alpha_i, \beta_i) \notin M$, x^* must be in (x_i, x_{i+1}) and we have

$$(4.2) \qquad S_0'(x^*) = \phi(\alpha_i, \beta_i)\delta_i.$$

We let

(4.3) $$\omega := S_0'(x^*), \quad \mu := x^* - x_i, \quad \eta := x_{i+1} - x^*.$$

Then we have $\mu + \eta = \Delta x_i$. We note that in the case under consideration, $\omega \delta_i < 0$ and therefore also $\omega s_i \leq 0$ and $\omega s_{i+1} \leq 0$. To go further, we assume $\omega < 0$. The case $\omega > 0$ is handled similarly. Now, if $s_i > s_{i+1}$ then (4.1) shows that $\mu > \Delta x_i/2 > \eta$. Similarly, if $s_{i+1} > s_i$ then $\eta > \Delta x_i/2 > \mu$. We rewrite Q in terms of x^* and ω; then for some constant a we have

(4.4) $$Q(x) = a(x - x^*)^2 + \omega,$$

(4.5) $$s_i = a\mu^2 + \omega, \qquad s_{i+1} = a\eta^2 + \omega.$$

Let ξ_{i1}, ξ_{i2} be the new knots which we will insert: $x_i < \xi_{i1} \leq x^* \leq \xi_{i2} < x_{i+1}$.

Our new interpolant S will be taken to have a derivative S' of the following form on I_i:

(4.6) $$S' = \begin{cases} Q_1(x) = a_1(x - \xi_{i1})^2, & x_i \leq x \leq \xi_{i1}, \\ 0, & \xi_{i1} \leq x \leq \xi_{i2}, \\ Q_2(x) = a_2(x - \xi_{i2})^2, & \xi_{i2} \leq x \leq x_{i+1}. \end{cases}$$

Before proceeding to describe the parameters in (4.6), it will be useful to say a few words about the form of S'. We have chosen to define S' to be zero between the knots ξ_{i1} and ξ_{i2}. This means that S will be constant on this interval, and therefore the graph will have a flat spot on this interval. We could have taken S' to be constant (or even more generally linear) on this interval but at the expense of making the algorithm more complicated.

We shall now spell out how we will choose the new knots ξ_{i1} and ξ_{i2} and the constants a_1 and a_2. There are actually infinitely many choices, since our only constraints are that S interpolates our data and the knots be in (x_i, x_{i+1}). To achieve the interpolation conditions we need

(4.7) $$\int_{x_i}^{x_{i+1}} S' = y_{i+1} - y_i = \int_{x_i}^{x_{i+1}} Q.$$

Using the form of S' given in (4.6), the first equality in (4.7) gives

(4.8) $$(\xi_{i1} - x_i)s_i + (x_{i+1} - \xi_{i2})s_{i+1} = 3(y_{i+1} - y_i).$$

Similarly, starting with (4.4) and using (4.3) and (4.5), the second equality in (4.7) gives

(4.8)' $$(s_i + 2\omega)\mu + (s_{i+1} + 2\omega)\eta = 3(y_{i+1} - y_i).$$

We choose ξ_{i1} and ξ_{i2} to satisfy (4.8) by letting

(4.9) $$\xi_{i1} = x_i + \frac{3\mu(y_{i+1} - y_i)}{s_i\mu + s_{i+1}\eta},$$

and

(4.10) $$\xi_{i2} = x_{i+1} - \frac{3\eta(y_{i+1} - y_i)}{s_i\mu + s_{i+1}\eta}.$$

There are other choices of knots which would result in fourth-order convergence, but the above choice, suggested by the referee, will depend continuously on the data. We now show that

(4.11) $$x_i < \xi_{i1} \leqslant x^* \leqslant \xi_{i2} \leqslant x_{i+1}.$$

For example, if (4.9) holds, then the left outermost inequality is clear since the second term on the right of (4.9) is positive. Also, using (4.8)', we have

$$x^* - \xi_{i1} = (s_i\mu + s_{i+1}\eta - 3\Delta y_i)\mu/(s_i\mu + s_{i+1}\eta)$$
$$= -2\omega(\mu + \eta)/(s_i\mu + s_{i+1}\eta) \geqslant 0$$

because $\omega < 0$ when $(\alpha_i, \beta_i) \notin M_i$. Therefore, $\xi_{i1} \leqslant x^*$ which is the second inequality in (4.11). The other inequalities in (4.11) follow from a similar analysis for the case (4.10).

For our algorithm we need an initial way of assigning slopes at x_i, $i = 0, \ldots, n$. For this, we let $d_i := d_i(f)$, $i = 1, \ldots, n - 2$, be an approximation to $f'(x_i)$ defined by a four-point formula which uses the values of f at x_{i-1}, x_i, x_{i+1}, x_{i+2}. That is,

(4.12)
$$d_i = \delta_{i-1} + \frac{\delta_i - \delta_{i-1}}{\Delta x_i + \delta x_{i-1}}\Delta x_{i-1}$$
$$+ \frac{(\delta_i - \delta_{i-1})/(\Delta x_i + \Delta x_{i-1}) - (\delta_{i+1} - \delta_i)/(\Delta x_{i+1} + \Delta x_i)}{\Delta x_{i+1} + \Delta x_i + \Delta x_{i-1}}\Delta x_{i-1}\Delta x_i.$$

For $i = 0$, $n - 1$, n we take d_i again to be a four-point formula which uses values of f at the four nearest points to x_i. Recall that a four-point formula computes derivatives exactly for any cubic polynomial U, i.e., $d_i(U) = U'(x_i)$, $i = 0, \ldots, n$.

We can summarize our algorithm in the following steps.

Step 1. Initially set $s_i = d_i$, $i = 0, \ldots, n$.

Step 2. We make sure that no two of s_i, s_{i+1}, δ_i are of opposite signs. That is, if $d_i\delta_i \leqslant 0$, we set $s_i = 0$ and if $d_{i+1}\delta_i \leqslant 0$ we set $s_{i+1} = 0$. Otherwise we still have $s_i = d_i$.

Step 3. For each interval I_i in which $(\alpha_i, \beta_i) \in M$, S can be represented as in (2.1) and S is monotone on I_i.

Step 4. If $(\alpha_i, \beta_i) \notin M$, we choose knots ξ_{i1}, ξ_{i2} by using (4.9), (4.10) and define S by

(4.13) $$S := \begin{cases} a_1(x - \xi_{i1})^3/3 + b & \text{on } [x_i, \xi_{i1}], \\ b & \text{on } [\xi_{i1}, \xi_{i2}], \\ a_2(x - \xi_{i2})^3/3 + b & \text{on } [\xi_{i2}, x_{i+1}], \end{cases}$$

with

$$a_1 := s_i/(x_i - \xi_{i1})^2, \qquad a_2 := s_{i+1}/(x_{i+1} - \xi_{i2})^2,$$
$$b := y_i - s_i(x_i - \xi_{i1})/3 = y_{i+1} - s_{i+1}(x_{i+1} - \xi_{i2})/3.$$

Then again S is monotone on I_i.

5. Convergence Order. We shall show that the algorithm described in Section 4 gives a fourth-order approximation to monotone functions. We discuss only the case when f is nondecreasing. Let $\|\cdot\|$ denote the supremum norm on $[0, 1]$. We consider the linear functionals $d_i(f) := d_i$ with d_i as in Section 4 (see (4.12)). Note that

when U is a cubic polynomial, we have $d_i(U) = U'(x_i)$, $i = 0, \ldots, n$. Now let f be four times continuously differentiable with $\|f^{(4)}\| = M$. Calculating the divided difference of f and using (4.12), we find for $1 \leqslant i \leqslant n - 2$

$$
(5.1) \quad |f'(x_i) - d_i| = \Delta x_i \Delta x_{i-1}(\Delta x_i + \Delta x_{i+1})|[x_{i-1}, x_i, x_i, x_{i+1}, x_{i+2}]f|
$$

$$
\leqslant Mh^3/12.
$$

In the same way, this holds for the cases $i = 0, n - 1, n$ as well.

LEMMA 5.1 [3, Lemma 3.2]. *If* $f \in C^1[0, 1]$ *is monotone nondecreasing, then, for the slope assignments* s_i *of Step* 2, *we have*

$$
(5.2) \qquad s_i \geqslant 0 \quad and \quad |f'(x_i) - s_i| \leqslant |f'(x_i) - d_i|, \qquad i = 0, \ldots, n.
$$

LEMMA 5.2 [6, Theorem 3.6]. *If* $H(x)$ *is the Hermite cubic interpolation to* $f \in C^{(4)}[0, 1]$, *then*

$$
\|f' - H'\| \leqslant Mh^3/6 \quad on \ [0, 1]
$$

with $M := \|f^{(4)}\|$ *and* $h =: \max_{0 \leqslant i < n} \Delta x_i$.

THEOREM 5.1. *If* f *is four times continuously differentiable and monotone on* $[0, 1]$, *then the algorithm of Section* 4 *generates a piecewise cubic* S *satisfying*

$$
(5.3) \qquad\qquad\qquad \|f - S\| \leqslant 9Mh^4.
$$

Proof. Let S_0 be the spline generated by our algorithm after Step 2 has been implemented, and let H be the Hermite cubic spline interpolant to f Then $S_0(x_i) = H(x_i)$, $i = 0, \ldots, n$, and by (5.1) and Lemma 5.1,

$$
(5.4) \qquad\qquad\qquad |H'(x_i) - S_0'(x_i)| \leqslant Mh^3/12.
$$

Since $H - S_0$ is a cubic polynomial on $[x_i, x_{i+1}]$, we have

$$
(5.5) \quad
\begin{aligned}
H(x) - S_0(x) &= \frac{H'(x_i) - S_0'(x_i)}{(x_{i+1} - x_i)^2}(x - x_i)(x_{i+1} - x)^2 \\
&\quad + \frac{H'(x_{i+1}) - S_0'(x_{i+1})}{(x_{i+1} - x_i)^2}(x - x_i)^2(x - x_{i+1}).
\end{aligned}
$$

From (5.5) and (5.4) we get $|H(x) - S_0(x)| \leqslant Mh^4/6$. Using the well-known error estimate for $f - H$ (see [2, p. 53]), this gives

$$
(5.6) \qquad\qquad\qquad |f(x) - S_0(x)| \leqslant Mh^4/3.
$$

We also note that if we take the derivative of both sides of (5.5) and use (5.4) again, we have

$$
(5.7) \qquad\qquad\qquad |H'(x) - S_0'(x)| \leqslant Mh^3/2.
$$

This, and Lemma 5.2, give

$$(5.8) \qquad |f'(x) - S_0'(x)| \leqslant Mh^3.$$

Now let S be the spline generated by our algorithm. We shall show that

$$(5.9) \qquad \|S' - S_0'\| \leqslant 8Mh^3.$$

Integrating and using the fact that $S - S_0$ vanishes at x_i, for $i = 0, \ldots, n$, gives $\|S - S_0\| \leqslant 8Mh^4$. This, together with (5.6), then proves the theorem.

We need only check (5.9) on intervals $[x_i, x_{i+1}]$ where Step 4 has been implemented. On such an interval, S' is given by (4.6) and $S_0' = Q$ is given by (4.4). We shall show below that, with ω defined in (4.3),

$$(5.10) \qquad \|S' - S_0'\|_{[x_i, x_{i+1}]} \leqslant 8|\omega|.$$

Now f is monotone on $[x_i, x_{i+1}]$ but S_0 is not; then $S_0'(x^*)f'(x^*) \leqslant 0$. Hence, by (5.8),

$$|\omega| = |S_0'(x^*)| \leqslant |S_0'(x^*) - f'(x^*)| \leqslant Mh^3,$$

and assuming (5.10), this gives (5.9). We prove (5.10) only for the interval $J := [x_i, x^*]$; the same estimate holds on $[x^*, x_{i+1}]$ as well. Now (5.10) is obvious when $s_i \leqslant 4|\omega|$, since then both $|S_0'|$ and $|S'|$ are less than or equal to $4|\omega|$ on J. Therefore, we can assume that $s_i \geqslant 4|\omega|$.

Let Q_1 be as in (4.6) and x_0 be the vertex of the parabola $y = Q(x) - Q_1(x)$. Then x_0 satisfies the equation

$$a(x_0 - x^*) = a_1(x_0 - \xi_{i1}) =: \theta.$$

Also (see (4.4), (4.6)),

$$(5.11) \qquad \begin{aligned} |Q(x_0) - Q_1(x_0)| &\leqslant |\omega| + |\theta[(x_0 - x^*) - (x_0 - \xi_{i1})]| \\ &\leqslant |\omega| + |\theta||\xi_{i1} - x^*|. \end{aligned}$$

Now if $x_0 \in [x_i, x^*]$ then $|\theta| = |a(x_0 - x^*)| \leqslant |a\mu| \leqslant (s_i + |\omega|)/\mu$.
We claim therefore that

$$(5.12) \qquad |Q(x_0) - Q_1(x_0)| \leqslant 8|\omega|.$$

Indeed, $x^* - \xi_{i1} = \mu - (\xi_{i1} - x_i)$, and therefore, if ξ_{i1} is chosen by (4.9), we have $|x^* - \xi_{i1}| = |\mu(s_i\mu + s_{i+1}\eta - 3\Delta y_i)|/(s_i\mu + s_{i+1}\eta)$, and thus, by (4.8)',

$$(5.13) \qquad |x^* - \xi_{i1}| = \mu|2\omega|(\mu + \eta)/(s_i\mu + s_{i+1}\eta).$$

If $s_i < s_{i+1}$, we can replace s_{i+1} by s_i in (5.13) and obtain $|x^* - \xi_{i1}| \leqslant 2\mu|\omega|/s_i$. While if $s_i > s_{i+1}$, it follows from (4.1) that $\mu = x^* - x_i > x_{i+1} - x^* = \eta$, and hence $\mu + \eta \leqslant 2\mu$. Therefore, if we replace s_{i+1} by zero and replace η by μ in (5.13),

we obtain $|x^* - \xi_{i1}| \leqslant 4\mu|\omega|/s_i$. Using this estimate for $|x^* - \xi_{i1}|$ together with our estimate for θ in the line after (5.11), we have

$$|Q(x_0) - Q_1(x_0)| \leqslant |\omega| + (s_i + |\omega|)/\mu \cdot 4\mu|\omega|/s_i$$

$$= |\omega| + \left(4|\omega| + 4|\omega|^2/s_i\right).$$

This gives (5.12) because $s_i \geqslant 4|\omega|$.

Returning now to the problem of estimating $S' - S_0'$ on J, we note that the $\max|S' - S_0'|$ on J is taken at one of the three points x_i, x_0 or x^*. But at these points x, $|S'(x) - S_0'(x)|$ is smaller than 0, $8|\omega|$ and $|\omega|$, respectively; thus we have shown (5.10). □

6. Numerical Examples. We consider now the graphs generated by our algorithm for two of the typical data sets considered in the literature.

AKIMA DATA [1]:
Here $n = 11$, and the x's and y's are given by

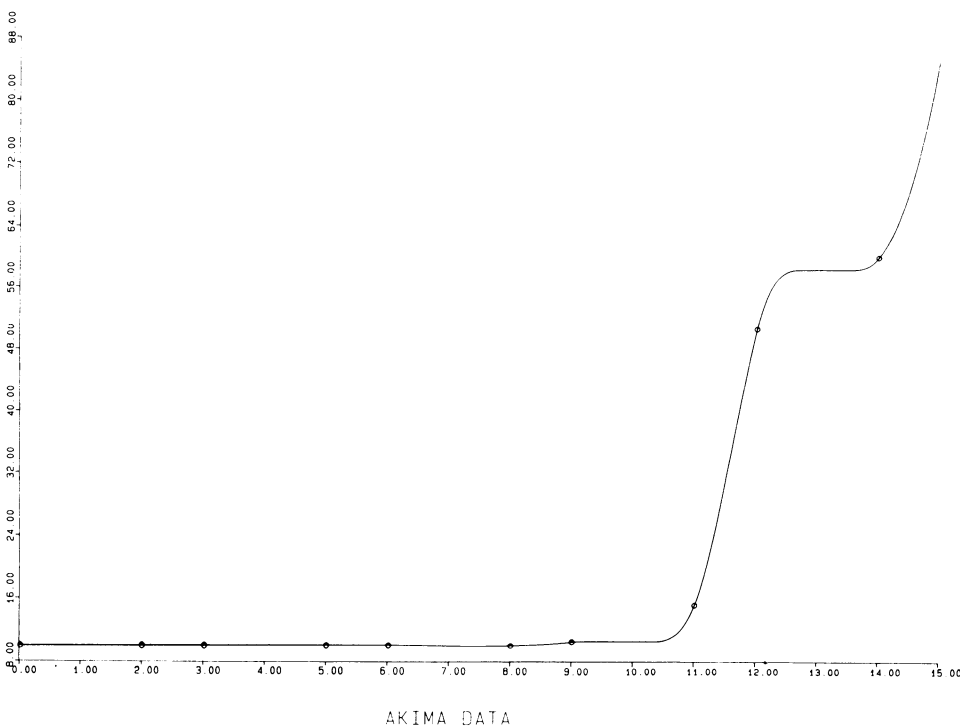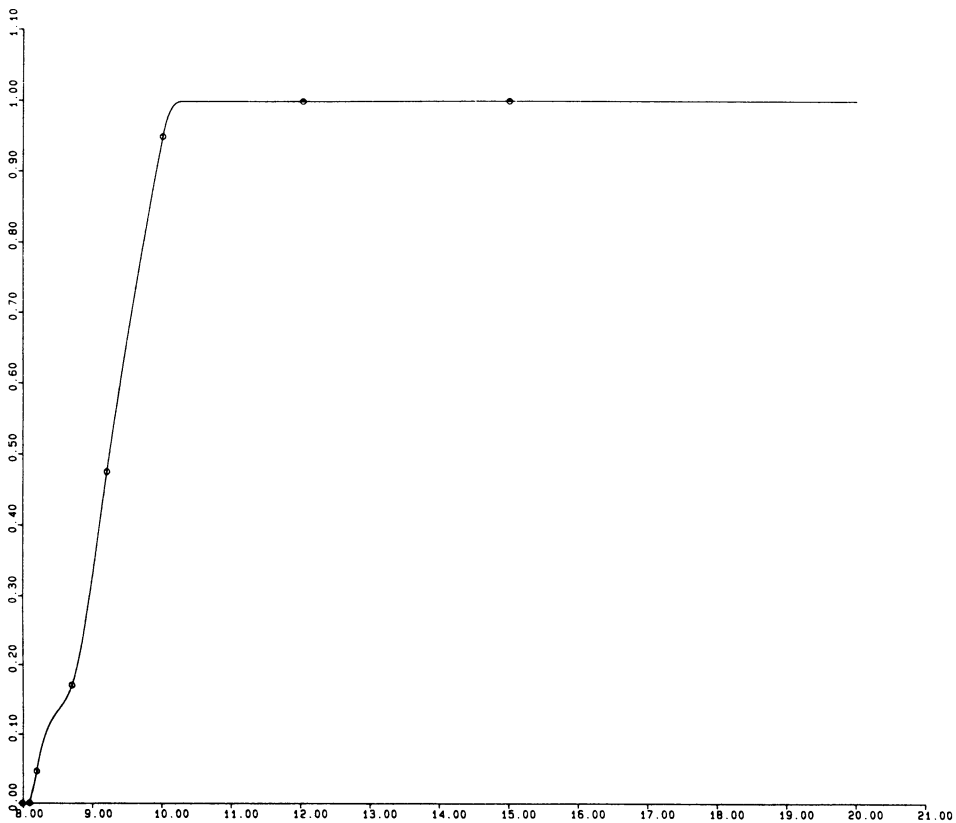| x	0	2	3	5	6	8	9	11	12	14	15
y	10	10	10	10	10	10	10.5	15	50	60	85

AKIMA DATA

FIGURE 1

Spline of 4 for AKIMA data

RNP 14 DATA [4]:

Here $n = 9$, and the x's and y's are given by

x	7.99	8.09	8.19	8.7	9.2	10
y	0	2.76429e–5	4.37498e–2	0.169183	0.469428	0.943740

x	12	15	20
y	0.998636	0.999919	0.999994

RNP 14 DATA

FIGURE 2

Spline of 4 for RNP 14 data

For these two examples, we see that the algorithm produces a pleasing interpolant when the slopes for the data do not change abruptly from a large to a small value. Near such an abrupt change however, the graph produced by our algorithm also changes quickly (apparently due to the high order of convergence) and is not as pleasing as those produced by some of the other standard algorithms (see, e.g., the sample graphs for the same data in [4]).

School of Mathematics
Georgia Institute of Technology
Atlanta, Georgia 30332

1. H. AKIMA, "A new method of interpolation and smooth curve fitting based on local procedures," *J. Assoc. Comput. Mach.*, v. 17, 1970, pp. 589–602.

2. C. DE BOOR, "A practical guide to splines," *Applied Mathematical Sciences*, Vol. 27, Springer-Verlag, Berlin and New York, 1978.

3. S. C. EISENSTAT, K. R. JACKSON & J. W. LEWIS, "The order of monotone piecewise cubic interpolation," *SIAM J. Numer. Anal.*, v. 22, 1985, pp. 1220–1237.

4. F. N. FRITSCH & R. E. CARLSON, "Monotone piecewise cubic interpolation," *SIAM J. Numer. Anal.*, v. 17, 1980, pp. 238–246.

5. D. F. MCALLISTER & J. A. ROULIER, "An algorithm for computing a shape-preserving osculatory quadratic spline," *ACM Trans. Math. Software*, v. 7, 1981, pp. 331–347.

6. M. H. SCHULTZ, *Spline Analysis*, Prentice-Hall, Englewood Cliffs, N. J., 1973.