Journal of Computational and Applied Mathematics 39 (1992) 287–294
North-Holland

Monotonic piecewise cubic interpolation, with applications to ODE plotting

D.J. Higham *

Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, M5S 1A4

Received 21 August 1990
Revised 3 March 1991

Abstract

Higham, D.J., Monotonic piecewise cubic interpolation, with applications to ODE plotting, Journal of Computational and Applied Mathematics 39 (1992) 287–294.

Given a set of solution and derivative values, we examine the problem of constructing a piecewise cubic interpolant which reflects the monotonicity present in the data. Drawing on the theory of Fritsch and Carlson (1980), we derive a simple algorithm that, if necessary, adds one or two extra knots between existing knots in order to preserve monotonicity. The new algorithm is completely local in nature and does not perturb the input data. We show that the algorithm is particularly suited to the case where the data arises from the discrete approximate solution of an ODE.

Keywords: Cubic polynomial, Hermite, interpolation, monotonicity, initial-value problem.

1. Introduction

Piecewise cubic Hermite interpolation is a popular method for fitting a continuously differentiable curve through a discrete set of solution and derivative data. One minor defect in this approximation technique is that monotonicity present in the data will not necessarily be reflected by the interpolant. For this reason several authors have derived algorithms which "massage" the derivative data, thereby guaranteeing that monotonicity will be preserved [1,4,6,7,10]. In this work, we present an alternative algorithm which, if necessary, adds one or two new knots in each subinterval. The resulting technique is completely local and does not alter the input data. We argue that this approach is particularly suited to the special case where the numerical solution of an ODE is to be plotted. The relevant theory of Fritsch and Carlson [7] is introduced in the next section, and in Section 3 we derive the new algorithm. The subproblem of plotting ODE data is discussed briefly in Section 4.

* Supported by the Information Technology Research Centre of Ontario and the Natural Science and Engineering Research Council of Canada. Present address: Department of Mathematics and Computer Science, University of Dundee, Dundee, DD1 4HN, United Kingdom.

2. The results of Fritsch and Carlson

Suppose we are given a set of knots $x_1 < x_2 < \cdots < x_n$ and corresponding function and derivative data $\{y_i\}_{i=1}^n$ and $\{d_i\}_{i=1}^n$. We may then define the unique piecewise cubic Hermite interpolant $p(x) \in C^1[x_1, x_n]$ that satisfies

(i) $p(x_i) = y_i$, $1 \leqslant i \leqslant n$;
(ii) $p'(x_i) = d_i$, $1 \leqslant i \leqslant n$;
(iii) $p(x)$ is a cubic polynomial on $[x_i, x_{i+1}]$, $1 \leqslant i \leqslant n-1$.

On the interval $[x_i, x_{i+1}]$ we denote the secant slope by $\Delta_i := (y_{i+1} - y_i)/(x_{i+1} - x_i)$, and we say that the data is monotone on this interval if

$$\frac{d_i}{\Delta_i} \geqslant 0, \qquad \frac{d_{i+1}}{\Delta_i} \geqslant 0,$$

when $\Delta_i \neq 0$ or if $\Delta_i = d_i = d_{i+1} = 0$. In many applications it is desirable that monotonicity of the data should be reflected in the interpolant. Necessary and sufficient conditions for $p(x)$ to be monotonic on $[x_i, x_{i+1}]$ were determined by Fritsch and Carlson [7] and are summarised in the following lemma.

Lemma. *If $\Delta_i = d_i = d_{i+1} = 0$, then $p(x)$ is monotonic on $[x_i, x_{i+1}]$. (In this case $p(x)$ reduces to a constant function.) If $\Delta_i \neq 0$, then letting*

$$\alpha_i = \frac{d_i}{\Delta_i}, \qquad \beta_i = \frac{d_{i+1}}{\Delta_i},$$

$p(x)$ is monotonic on $[x_i, x_{i+1}]$ if and only if $(\alpha_i, \beta_i) \in \mathscr{M}$, where the monotonicity region $\mathscr{M}$ is shown in Fig. 2.1. Formally, $\mathscr{M} := \{(\alpha, \beta) \mid \phi(\alpha, \beta) \geqslant 0\} \cup \{(\alpha, \beta) \mid 0 \leqslant \alpha \leqslant 3,\ 0 \leqslant \beta \leqslant 3\}$, where $\phi(\alpha, \beta) = \alpha - (2\alpha + \beta - 3)^2/(3(\alpha + \beta - 2))$.

Note that for $\Delta_i \neq 0$, the condition for the data to be monotonic is that (α_i, β_i) lies in the first quadrant of $\mathbb{R}^2$. Hence it is clear that there exists monotonic data for which the corresponding cubic Hermite interpolant is not monotonic (for example, if $\alpha_i > 4$ or $\beta_i > 4$). To

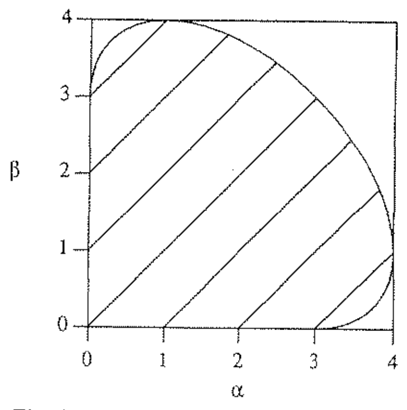

Fig. 2.1. The monotonicity region $\mathscr{M}$.

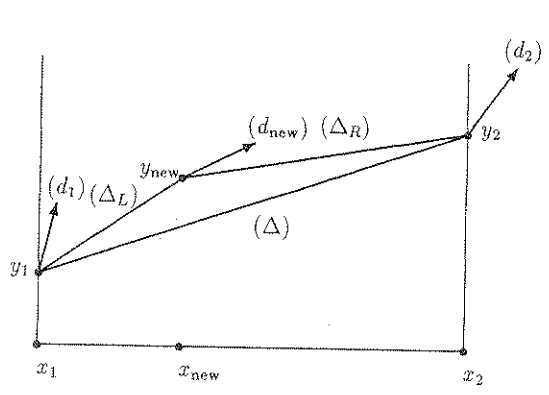

Fig. 3.1. Quantities in parentheses are tangent slopes.

avoid this problem, Fritsch and Carlson proposed an algorithm for perturbing the derivative data d_i so that the monotonicity of the data is unchanged, and is reproduced in the interpolant. To apply the algorithm, a region $\mathscr{T} \subset \mathscr{M}$ satisfying

$$(\alpha, \beta) \in \mathscr{T} \Rightarrow (\alpha^*, \beta^*) \in \mathscr{T}, \quad \text{for } 0 \leqslant \alpha^* \leqslant \alpha,\ 0 \leqslant \beta^* \leqslant \beta,$$

must be chosen. The algorithm ensures that the new data $\{x_i, y_i, d_i^*\}$ satisfies $\{d_i^*/\Delta_i, d_{i+1}^*/\Delta_i\} \in \mathscr{T}$, $1 \leqslant i \leqslant n-1$.

Fritsch and Carlson had in mind the application where the derivative data $\{d_i\}$ is not supplied directly, but rather is computed from the function data $\{y_i\}$ by the use of, for example, three point difference formulas. In this case it is natural to consider altering the $\{d_i\}$ values in order to produce a monotonic cubic spline. We point out that the algorithm may "unnecessarily" perturb the data, since it ensures that the (α_i, β_i) pairs lie in $\mathscr{T}$ rather than $\mathscr{M}$. Alternative techniques for perturbing the d_i values have since been proposed in [1,4,6,10].

In this work, we are concerned with the application where the data arises from the discrete solution of an initial-value ODE problem

$$y'(x) = f(x, y(x)), \quad y(a) = y_a,\ a \leqslant x \leqslant b.$$

Here we are given approximations $y_i \approx y(x_i)$ and $d_i := f(x_i, y_i) \approx y'(x_i)$. (In the case of a system of ODEs we consider each component of the solution separately.) In this situation we will generally have an equal amount of confidence in the solution data y_i and the derivative data d_i, since if f satisfies a Lipschitz condition, the asymptotic order of accuracy of d_i is the same as that of y_i. Hence it seems undesirable to change the derivative values. Note that altering d_i also relaxes the condition that the residual $p'(x) - f(x, p(x))$ be zero at the knots.

The main purpose of this work is to present an alternative algorithm for preserving monotonicity in cubic splines. With this algorithm, if the original cubic Hermite does not preserve monotonicity in an interval, then we insert one or two extra knots, with corresponding solution and derivative data. As pointed out by Fritsch and Carlson, two disadvantages of adding knots are that the amount of extra storage required for the data, and the amount of search time needed to evaluate the spline will both be increased. However, as we will see in Section 4, the percentage of intervals $[x_i, x_{i+1}]$ on which the cubic Hermite fails to preserve monotonicity is typically very small in the case of ODE data. Also, the storage requirements of the original spline cannot be determined a priori, since the number of knots n depends on both the differential equation and the accuracy requirement.

We mention that other shape-preserving techniques which use the idea of adding knots have appeared in the literature; see, for example, [1,11]. In the former reference, by altering the derivative data the $\{\alpha_i, \beta_i\}$ values are forced into a region which contains $\mathscr{M}$, and then, if necessary, an extra knot is added in each interval. (The algorithm proposed here differs in that it is completely local and does not change the $\{d_i\}$ data.)

3. An algorithm for adding knots

We now develop an algorithm for adding knots in those intervals where the data is monotone, but the corresponding cubic polynomial is not. For simplicity of notation, we label the data $\{x_1, y_1, d_1; x_2, y_2, d_2\}$ and we let

$$\Delta = \frac{y_2 - y_1}{x_2 - x_1}, \qquad \alpha = \frac{d_1}{\Delta}, \qquad \beta = \frac{d_2}{\Delta}.$$

(We may assume that $\Delta \neq 0$, since if $\Delta = 0$, then the cubic Hermite is monotone if and only if the data is monotone.) For the moment we assume that the data is monotonic increasing with $\alpha \geqslant \beta$. If $(\alpha, \beta) \notin \mathscr{M}$, then we hope to add a new knot, x_{new}, with corresponding data y_{new} and d_{new} so that $y_1 < y_{\text{new}} < y_2$, $d_{\text{new}} \geqslant 0$ and the cubic Hermites on $[x_1, x_{\text{new}}]$ and $[x_{\text{new}}, x_2]$ are both monotonic increasing. Figure 3.1 illustrates the situation. We let Δ_{L} and Δ_{R} denote the secant slopes on the left and right intervals; that is,

$$\Delta_{\text{L}} = \frac{y_{\text{new}} - y_1}{x_{\text{new}} - x_1}, \qquad \Delta_{\text{R}} = \frac{y_2 - y_{\text{new}}}{x_2 - x_{\text{new}}},$$

and we write $x_{\text{new}} = x_1 + rI$, where $I = x_2 - x_1$, and $0 < r < 1$. The new data is completely specified by Δ_{L}, r and d_{new}. We also define

$$\alpha_{\text{L}} = \frac{d_1}{\Delta_{\text{L}}}, \qquad \beta_{\text{L}} = \frac{d_{\text{new}}}{\Delta_{\text{L}}}, \qquad \alpha_{\text{R}} = \frac{d_{\text{new}}}{\Delta_{\text{R}}}, \qquad \beta_{\text{R}} = \frac{d_2}{\Delta_{\text{R}}}$$

to be the alpha and beta values on the new left and right intervals. We aim to make $(\alpha_{\text{L}}, \beta_{\text{L}}) \in \mathscr{M}$ and $(\alpha_{\text{R}}, \beta_{\text{R}}) \in \mathscr{M}$. Since $(\alpha, \beta) \notin \mathscr{M}$ and $\alpha \geqslant \beta$, we must have $\alpha > 3$ (see Fig. 2.1). In other words, the derivative value d_1 is more than three times as large as the secant slope Δ. To alleviate this, we must choose $\Delta_{\text{L}} > \Delta$, say $\Delta_{\text{L}} = K\Delta$ for some $K > 1$. For reasons which will become clear shortly, we choose $K > \frac{1}{3}\alpha$, so that $\alpha_{\text{L}} < 3$. The monotonicity condition, $y_{\text{new}} < y_2$, then implies that r must be chosen to satisfy $r < 1/K$. From Fig. 3.1 we see that $\Delta_{\text{R}} < \Delta$ and hence $\beta_{\text{R}} > \beta$. Since $(\alpha_{\text{R}}, \beta_{\text{R}}) \in \mathscr{M} \Rightarrow \beta_{\text{R}} \leqslant 4$, we cannot hope for the algorithm to succeed unless $\beta < 4$. Assuming $\beta < 4$, and using the fact that

$$\beta_{\text{R}} = \beta \frac{(1-r)}{1 - rK},$$

it follows that by choosing $r < (4 - \beta)/(4K - \beta)$ we force $\beta_{\text{R}} < 4$. (Notice that this represents a stronger restriction on r than the monotonicity constraint $r < 1/K$.) It is clear from Fig. 2.1 that given any $\beta_{\text{R}} < 4$ there exists a value $\alpha_{\text{R}} \in [1, 3]$ such that $(\alpha_{\text{R}}, \beta_{\text{R}}) \in \mathscr{M}$. After choosing such a value for α_{R}, we set $d_{\text{new}} = \alpha_{\text{R}}\Delta_{\text{R}} = \alpha_{\text{R}}\Delta(1 - rK)/(1 - r)$. Since $\Delta_{\text{R}} < \Delta_{\text{L}}$, we have $\beta_{\text{L}} = d_{\text{new}}/\Delta_{\text{L}} < d_{\text{new}}/\Delta_{\text{R}} = \alpha_{\text{R}} \leqslant 3$. Finally, since we have forced $\alpha_{\text{L}} < 3$, it follows that $(\alpha_{\text{L}}, \beta_{\text{L}}) \in \mathscr{M}$.

The resulting algorithm for adding a knot is straightforward:

(i) choose $K > \frac{1}{3}\alpha$, and $r < (4 - \beta)/(4K - \beta)$;
(ii) choose $\alpha_{\text{R}} \in [1, 3]$ such that $(\alpha_{\text{R}}, \beta(1 - r)/(1 - rK)) \in \mathscr{M}$;
(iii) let $x_{\text{new}} = x_1 + rI$, $y_{\text{new}} = y_1 + rIK\Delta$, $d_{\text{new}} = \alpha_{\text{R}}\Delta(1 - rK)/(1 - r)$.

The search in (ii) could be performed, for example, by decreasing α_{R} from 3 to 1 in steps of 0.1. It is guaranteed to finish at, or before, $\alpha_{\text{R}} = 1$. It can be shown that this algorithm also handles monotonic decreasing data for which $\alpha \geqslant \beta$ and $\beta < 4$. A corresponding technique for the case $\beta > \alpha$ and $\alpha < 4$ can be derived in a similar way. The two versions, with specific choices for K and r, are summarised as Algorithm 1.

Algorithm 1. For the case $\min\{\alpha, \beta\} < 4$.
(Generates a new knot in order to preserve monotonicity.)

Algorithm 1A. For $\alpha \geqslant \beta$:
(i) let $K = 1.1\alpha/3$ and $r = 0.8(4-\beta)/(4K-\beta)$;
(ii) choose $\alpha_R \in [1, 3]$ such that $(\alpha_R, \beta(1-r)/(1-rK)) \in \mathscr{M}$;
(iii) let $x_{new} = x_1 + rI$, $y_{new} = y_1 + rIK\Delta$, $d_{new} = \alpha_R \Delta(1-rK)/(1-r)$.

Algorithm 1B. For $\beta > \alpha$:
(i) let $K = 1.1\beta/3$ and $r = 0.8(4-\alpha)/(4K-\alpha)$;
(ii) choose $\beta_L \in [1, 3]$ such that $(\alpha(1-r)/(1-rK), \beta_L) \in \mathscr{M}$;
(iii) let $x_{new} = x_1 + (1-r)I$, $y_{new} = y_1 + (1-rK)I\Delta$, $d_{new} = \beta_L \Delta(1-rK)/(1-r)$.

It is clear from the derivation of Algorithm 1 that with $\min\{\alpha, \beta\} \geqslant 4$ it is not possible to preserve monotonicity by adding a single knot. However, we show below that in this case two extra knots will suffice. The basic idea is to pre-process the interval into the form required by Algorithm 1.

Again, for definiteness, we suppose that the data is monotonic increasing, and we refer to Fig. 3.1. Suppose now that $\alpha \geqslant \beta \geqslant 4$. Our aim is to add the knot x_{new} and data y_{new}, d_{new} so that $(\alpha_L, \beta_L) \in \mathscr{M}$ and $\alpha_R \leqslant 3$. We may then apply Algorithm 1B to the interval $[x_{new}, x_2]$. It also seems desirable to avoid an unnecessarily large β_R value, so we will aim to keep $\beta_R \leqslant 2\beta$. As before, we will develop a method for choosing K, r and d_{new}. We may choose $\Delta_L = K\Delta$ where $K > \frac{1}{3}\alpha$, which ensures that $\alpha_L < 3$. The monotonicity constraint $y_{new} < y_2$ then reduces to $r < 1/K$. The additional constraint $\beta_R \leqslant 2\beta$ actually imposes the more severe restriction $r \leqslant 1/(2K-1)$. Finally, choose d_{new} so that $0 \leqslant \alpha_R \leqslant 3$, that is, $0 \leqslant d_{new} \leqslant 3\Delta(1-rK)/(1-r)$. Since $0 < \Delta_R < \Delta_L$, it follows that $\beta_L < \alpha_R \leqslant 3$, so $(\alpha_L, \beta_L) \in \mathscr{M}$ as required. This leads to the following strategy:
(i) choose $K > \frac{1}{3}\alpha$, and $r \leqslant 1/(2K-1)$;
(ii) let $x_{new} = x_1 + rI$, $y_{new} = y_1 + rIK\Delta$, $d_{new} = \theta 3\Delta(1-rK)/(1-r)$, for some $\theta \in [0, 1]$.
The method also works for monotonic decreasing data, and there is an analogous version for the case where $\beta \geqslant \alpha \geqslant 4$ which inserts a knot so that $(\alpha_R, \beta_R) \in \mathscr{M}$, $\beta_L \leqslant 3$ and $\alpha_L \leqslant 2\alpha$. Algorithm 2 gives the two versions, with specific choices for K, r and θ. (Note that the choice $r = 1/(2K-1)$ below makes $(1-rK)/(1-r) = 0.5$ in (ii).)

Algorithm 2. For the case $\min\{\alpha, \beta\} \geqslant 4$.
(Generates two new knots in order to preserve monotonicity.)

Algorithm 2A. For $\alpha \geqslant \beta$:
(i) let $K = 1.1\alpha/3$ and $r = 1/(2K-1)$;
(ii) let $x_{new} = x_1 + rI$, $y_{new} = y_1 + rIK\Delta$, $d_{new} = 0.8 \times 3\Delta \times 0.5$;
(iii) apply Algorithm 1B to $[x_{new}, x_2]$.

Algorithm 2B. For $\beta > \alpha$:
(i) let $K = 1.1\beta/3$ and $r = 1/(2K-1)$;
(ii) let $x_{new} = x_1 + (1-r)I$, $y_{new} = y_1 + (1-rK)I\Delta$, $d_{new} = 0.8 \times 3\Delta \times 0.5$;
(iii) apply Algorithm 1A to $[x_1, x_{new}]$.

As a practical point, we mention that it may be undesirable to insert a knot too close to an existing knot (in a relative sense). Since $r < 0.8$ in Algorithm 1, and $r < 0.52$ in Algorithm 2, x_{new} can only be close to one end of the current interval if r is small. In Algorithm 1 it is easy to show that $r \approx 0$ if either $\min\{\alpha, \beta\}$ is close to 4, or $\max\{\alpha, \beta\}$ is large. For example, in Algorithm 1A, we see from (i) that if ϵ is a small positive quantity, then

$$r = 0.8\epsilon \quad \Leftrightarrow \quad \frac{4-\beta}{4K-\beta} = \epsilon \quad \Leftrightarrow \quad \beta = \frac{4(1-\epsilon K)}{1-\epsilon} \approx 4 - 4\epsilon(K-1).$$

The difficulty caused by $\min\{\alpha, \beta\} \approx 4$ is easily overcome — switch to Algorithm 2, inserting two new knots rather than one. With Algorithm 2, a tiny r value can only arise if $\max\{\alpha, \beta\}$ is large. In such cases, the secant slope differs greatly from one of the derivative values, and hence the data may be regarded as somewhat inconsistent. In this situation the monotonicity constraint on y_{new} makes close knots unavoidable.

We conclude this section with two numerical examples. In both cases we interpolate over a single interval. The two sets of $\{x_i, y_i, d_i\}$ data that we use are Set $A = \{\{0, 1, 10\}, \{1, 3, 6\}\}$, for which $\alpha = 5$, $\beta = 3$, and Set $B = \{\{-3, 0, -6\}, \{-2, -1, -6.1\}\}$, for which $\alpha = 6$, $\beta = 6.1$. The data, which was chosen artificially, may be considered rather extreme in the sense that the cubic interpolants are far from monotone. On set A, Algorithm 1A adds a new knot at $x_{\text{new}} = 0.185$ with $y_{\text{new}} = 1.68$ and $d_{\text{new}} = 3.25$. On set B, Algorithm 2B gives $x_{\text{new}} = -2.29$, $y_{\text{new}} = -0.36$ and $d_{\text{new}} = -1.20$, and then Algorithm 1A gives $x_{\text{new}} = -2.94$, $y_{\text{new}} = -0.13$ and $d_{\text{new}} = -0.82$. "Before and after" plots of the cubic Hermite interpolants are presented in Figs. 3.2 and 3.3.

4. Discussion

In this section we focus on the case of shape-preserving interpolation for ODEs, and mention some related work [2,8]. In [2] Brankin and Gladwell used standard software to solve a range of stiff and nonstiff ODEs taken from the package [5]. Among the statistics that they

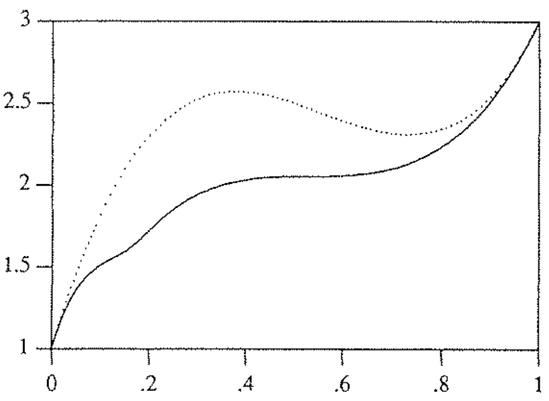

Fig. 3.2. Cubic Hermite interpolants for data set A. (Dotted line = before Algorithm 1A; solid line = after Algorithm 1A.)

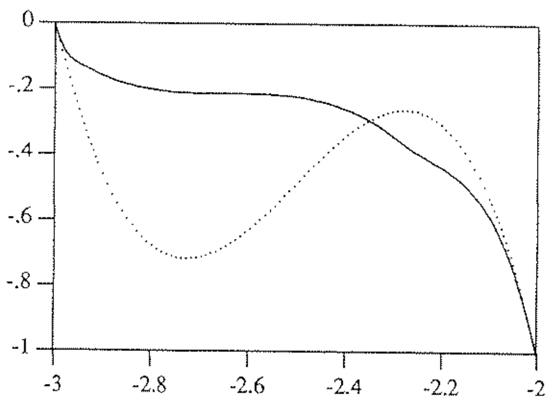

Fig. 3.3. Cubic Hermite interpolants for data set B. (Dotted line = before Algorithm 2B; solid line = after Algorithm 2B.)

Table 4.1
Proportion of intervals where cubic Hermite preserved monotonicity

Relative local error tolerance	10^{-1}	10^{-2}	10^{-3}	10^{-6}
Nonstiff test set	0.98	0.99	0.99	1.0
Stiff test set	0.92	0.94	0.96	0.98

recorded were the proportion of intervals $[x_i, x_{i+1}]$ on which the cubic Hermite interpolant preserved monotonicity. These values are reproduced (from the technical report [2] with permission) in Table 4.1. The fact that the proportion of "successful" intervals increases as the error tolerance decreases is not surprising. As the tolerance is reduced, the data $\{y_i, d_i\}$ becomes more accurate, and the corresponding cubic Hermite interpolant converges to the true solution, inheriting the same shape. However, even at the lax tolerances of 10^{-1} and 10^{-2} we see that the cubic Hermite interpolant would preserve monotonicity on the vast majority of steps.

A technique for preserving monotonicity in ODE interpolants is presented in [8]. This strategy makes use of the rational quadratic interpolant of [9] which interpolates to solution and derivative values and guarantees to preserve monotonicity in the data on each interval $[x_i, x_{i+1}]$. Gladwell et al. [8] recommend the use of the rational quadratic on intervals where the data is monotonic, and the cubic Hermite on the remaining intervals (since if the data is not monotonic, the rational quadratic may have a pole). Since the standard cubic Hermite interpolant is cheaper to use than the rational function [2, Section 4] and will be successful on most intervals, the technique proposed here of using one cubic per interval by default and two or three cubics per interval in the exceptional cases seems to be a reasonable alternative from the points of view of efficiency and ease of use.

Another important characteristic of an ODE interpolant is the local order of accuracy. Defining the local solution $u_i(x)$ over $[x_i, x_{i+1}]$ by

$$u_i'(x) = f(x, u_i(x)), \quad u_i(x_i) = y_i,$$

we say that an interpolant $p(x)$ has local order q if q is the largest integer such that

$$u_i(x_i + \tau h_i) - p(x_i + \tau h_i) = \mathrm{O}(h_i^q),$$

where $h_i = x_{i+1} - x_i$, for any fixed $\tau \in [0, 1]$. If the data $\{y_i, y_{i+1}, d_i, d_{i+1}\}$ comes from a sufficiently high-order integration method, then the cubic Hermite interpolant and the rational quadratic interpolant both have local order four. Theoretically, the algorithm presented here for adding knots will not alter the local order, since, for sufficiently small stepsizes h_i, the standard cubic Hermite will automatically preserve (strict) monotonicity [8, p.337]. More specifically, suppose that as the error tolerance tends to zero we have $y_i = y(x_i) + \mathrm{o}(h_i)$ and hence, for Lipschitzian f, $d_i := f(x_i, y_i) = f(x_i, y(x_i)) + \mathrm{o}(h_i) = y'(x_i) + \mathrm{o}(h_i)$. Since $\Delta_i = y'(x_i) + \mathrm{o}(1)$, it then follows that, for $y(x) \neq 0$, $\alpha_i = 1 + \mathrm{o}(1)$, and similarly $\beta_i = 1 + \mathrm{o}(1)$, ensuring that (α_i, β_i) ultimately lies inside $\mathscr{M}$.

We also mention that a more complicated, and more powerful shape-preserving algorithm for ODE interpolation is described in [2]. This scheme combines the cubic Hermite and the monotonicity/convexity preserving rational cubic interpolant of [3]. The combination provides a monotonicity- and convexity-preserving interpolant of local order four.

As a concluding remark, we point out that it would be useful to have shape-preserving interpolation schemes of local order greater than four, since many ODE methods achieve accuracy higher than $O(h_i^4)$ at the knots.

Acknowledgements

I thank Nick Higham and Ken Jackson for commenting on this manuscript.

References

[1] R.K. Beatson and H. Wolkowicz, Post-processing piecewise cubics for monotonicity, *SIAM J. Numer. Anal.* **26** (1989) 480–502.

[2] R.W. Brankin and I. Gladwell, Shape-preserving local interpolation for plotting solutions of ODEs, *IMA J. Numer. Anal.* **9** (1989) 555–566; (earlier version appeared as Numerical Analysis Report 132, Dept. Math., Univ. Manchester).

[3] R. Delbourgo and J.A. Gregory, Shape preserving piecewise rational interpolation, *SIAM J. Sci. Statist. Comput.* **6** (1985) 967–976.

[4] S.C. Eisenstat, K.R. Jackson and J.W. Lewis, The order of monotone piecewise cubic interpolation, *SIAM J. Numer. Anal.* **22** (1985) 1220–1237.

[5] W.H. Enright and J.D. Pryce, Two FORTRAN packages for assessing initial value methods, *ACM Trans. Math. Software* **13** (1987) 1–27.

[6] F.N. Fritsch and J. Butland, A method for constructing local monotone piecewise cubic interpolants, *SIAM J. Sci. Statist. Comput.* **5** (1984) 300–304.

[7] F.N. Fritsch and R.E. Carlson, Monotone piecewise cubic interpolation, *SIAM J. Numer. Anal.* **17** (1980) 238–246.

[8] I. Gladwell, L.F. Shampine, L.S. Baca and R.W. Brankin, Practical aspects of interpolation in Runge–Kutta codes, *SIAM J. Sci. Statist. Comput.* **8** (1987) 322–341.

[9] J.A. Gregory and R. Delbourgo, Piecewise rational quadratic interpolation to monotonic data, *IMA J. Numer. Anal.* **2** (1982) 123–130.

[10] J.M. Hyman, Accurate monotonicity preserving cubic interpolation, *SIAM J. Sci. Statist. Comput.* **4** (1983) 645–654.

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