Controllable Local Monotonic Cubic Interpolation in Fluid Animations

Insung Ihm, Deukhyun Cha, Byungkwon Kang
Department of Computer Science
Sogang University
Seoul, Korea

Abstract

While linear interpolation has been used frequently in computer graphics, higher-order interpolation is often desirable in applications requiring higher-order accuracy. In this paper, we study how interpolation filters, employed to resample such data as velocity, density, and temperature in simulating the equations of fluid dynamics, affect the animation of fluids. For this purpose, we have designed a *controllable* local cubic interpolation scheme that offers G^1 (or C^1) continuity globally. It is based on monotonic splines so does not suffer from undue overshooting. Furthermore, it is possible to control the general behavior of the interpolation through a global tension parameter, providing a continuous spectrum of linear to cubic interpolation. We analyze how this controllable interpolation filter may be effectively used to enhance the visual reality for physically based fluid animation.

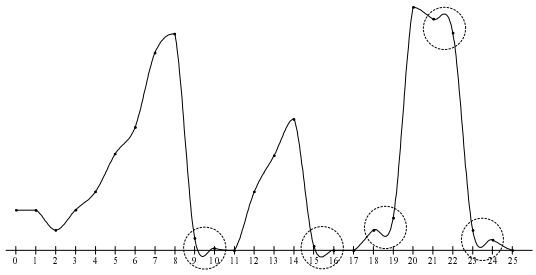
Keywords: Interpolation, cubic polynomial, monotonicity, controllability, fluid animation.

1 Introduction

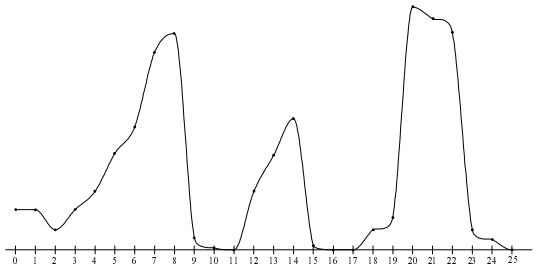
Reconstructing an original function from sampled data is a fundamental problem in computer graphics. Linear interpolation has frequently been used for this purpose because of its simplicity, but usually introduces significant reconstruction artifacts. A higher-order interpolation filter is often desirable in applications requiring high-order accuracy. Several high-order reconstruction filters, including piecewise cubic and windowed sinc filters, have been tested in attempts to improve the reconstruction quality [1, 2, 3, 4, 5]. While these filters have been considered too slow compared to linear interpolation, recent improvements in graphics hardware allow efficient hardware implementations of high-order filters [6].

This research was motivated by Fedkiw et al.'s work [7], where it was observed that cubic interpolation tends to reduce the amount of numerical dissipation and to generate more fine details in smoke simulation. One difficulty in employing a cubic interpolant is that it often suffers from numerical instabilities caused by undesirable oscillations. As Fedkiw et al. pointed out, a cubic interpolation scheme with overshoot must be avoided. Overshoot causes numerical instability and also produces severe aliases in animation and rendering. An interpolation filter with overshoot can also generate data values in improper ranges. For instance, when non-negative smoke densities are interpolated, negative values can be introduced, which should never happen.

A possible solution to avoid such undue overshoot is to design a monotonic interpolation filter. A monotonic filter gives rise to monotonic interpolating curves without oscillations for monotonic input data. Care must be taken in developing monotonic interpolation filters. In [7], a necessary condition was used to fix the overshooting problem as will be described later. Although it helps to reduce the overshooting problem, necessary conditions cannot fix it



(a) Before monotonicity is enforced



(b) After monotonicity is enforced with $\tau_{global} = 0.0$

Figure 1: An example of cubic interpolation. Generally, the Catmull–Rom spline generates very nice-looking curves. However, it often produces inappropriate overshoots even for monotone data points like those marked with dotted circles. When monotonicity is enforced, the oscillations go away.

completely. Figure 1(a) shows an example of the cubic Hermite interpolation obtained when monotonicity is not enforced properly. In this example, the slopes at the interpolation points except the extreme points were generated using the method employed by the Catmull–Rom spline algorithm. It is clear that a sequence of monotone data, for example one in the interval from 20 to 25, can easily produce non-monotonic splines with ugly overshoots. On the other hand, Figure 1(b) illustrates cubic interpolation splines produced using the technique presented here, where the inappropriate oscillations are nicely removed.

1.1 Related Work on Piecewise Monotonic Cubic Interpolation

While there is a large amount of literature in the area of monotonic interpolation, we focus on piecewise monotonic cubic interpolation in this paper. When only positional information $P = \{(x_i, y_i) \mid i = 0, 1, 2, \cdots, n\}$ is provided, which is usual in many applications, the key is the choice of appropriate first derivatives y_i' at the interpolation points x_i. The natural cubic spline technique determines them implicitly in such a way that the resulting splines are C^2 continuous [8]. Wolberg and Alfy applied the monotonicity constraints introduced by Fritsch and Carlson [9] to select derivatives that guarantee monotonic splines [10]. Because C^2 continuity is not always possible, they applied some optimization techniques to find the smoothest possible splines by minimizing the second derivative discontinuity.

Since Schweikert introduced exponential tension splines that allow the elimination of extraneous inflection points in cubic splines [11], tension parameters have frequently been employed to control the shape of both exponential and polynomial splines (refer to, for example, [12, 13] for extensive references). While these methods compute smooth curves, such global techniques are not well-suited to most graphics applications, where data should be reconstructed quickly using only local information. Local monotonic schemes were proposed in [14, 15, 16]. These schemes are based on C^1 continuous piecewise cubic Hermite interpolation using locally determined slopes, satisfying the monotonicity constraints given in [9]. They are quite simple to use, but on the other hand the splines they produce are often too restricted because they rely only on sufficient conditions. There is no flexibility in controlling the curve's shape because a fixed formula is applied to the selection of slopes. Manni also used the same monotonicity constraints to develop a local monotone interpolation scheme through parametric cubic splines [17].

1.2 Specific Contributions

The major motivation of this work is to provide the user with a continuous spectrum of linear to cubic monotonic interpolation filters, and to allow him/her to select a filter that produces the most desirable visual appearance. For this purpose, we first develop a *controllable* local monotonic cubic interpolation scheme that can be used effectively in fluid simulations. In particular, we attempt to design a cubic filter of the form $y^* =$

CUBIC_INTERP$(y_{-1}, y_0, y_1, y_2, x^*; \tau_{global})$, $0 \leq x^* \leq 1$, where sampled data $y(i) = y_i,\ i \in \{-1, 0, 1, 2\}$, are used to reconstruct function values $y = y(x)$ on $x \in [0, 1]$. In doing this, we propose to explore parametric cubic interpolation, which offers more flexibility than does functional interpolation. In our monotonic interpolation technique, the initial tangent vectors at the interpolation points are assigned using the Catmull–Rom spline technique. The monotonic constraints are then enforced, if necessary, by controlling the magnitude of the tangent vectors using local tension parameters at the data points.

As well as locality and monotonicity, the cubic interpolation scheme presented has the following additional properties:

- The interpolation is at least G^1 continuous (or C^1 continuous) globally. A four-point stencil as above is enough to achieve G^1 continuity, while a six-point stencil $\{y_{-2}, y_{-1}, \cdots, y_3\}$ is necessary to achieve C^1 continuity.

- The global shape of the cubic interpolant is controllable using a *global tension parameter* $\tau_{global} \in [0, 1]$. As the parameter increases from 0 to 1, the interpolation spline becomes gradually flattened, becoming a linear interpolation at $\tau_{global} = 1$. This parameter enables the user to control the general appearance of cubic interpolation in an intuitive fashion.

2 Preliminaries

2.1 The Cubic Hermite Interpolation

Throughout this paper, it is assumed that we want to reconstruct a function $y = y(x)$ on $[0, 1]$ using a cubic spline from a set of sampled data $\{y_i = y(i) \mid i \in \{-1, 0, 1, 2\}\}$ (see Figure 2). One of the most frequently used techniques is the cubic Hermite interpolation, which enforces the positional and tangential constraints: $y(i) = y_i$ and $y'(i) = y'_i$ $(i = 0, 1)$. The resulting cubic Hermite spline may be represented using the usual cubic Hermite basis functions as follows : $y_{[y_0, y_1, y'_0, y'_1]}(x) = (2x^3 - 3x^2 + 1)y_0 + (-2x^3 + 3x^2)y_1 + (x^3 - 2x^2 + x)y'_0 + (x^3 - x^2)y'_1$ $(0 \leq x \leq 1)$.

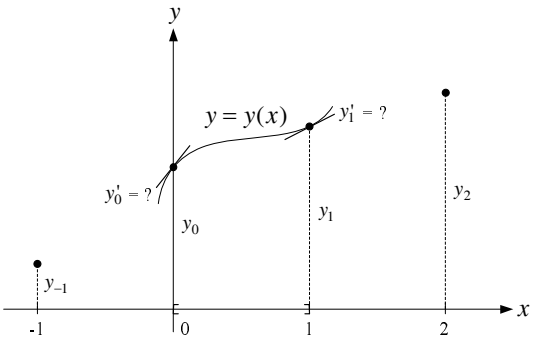


Figure 2: Cubic Hermite interpolation in functional form. A four-point stencil $\{-1, 0, 1, 2\}$ is used to reconstruct the functional value $y = y(x)$ on $x \in [0, 1]$.

(We drop the quadruple $[y_0, y_1, y'_0, y'_1]$ from the notation for simplicity unless its absence causes confusion.)

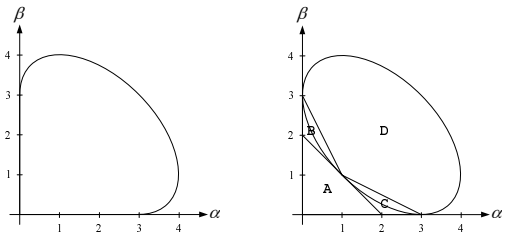
When this interpolation scheme is adopted, an important question is how to choose the derivative values y'_0 and y'_1. Various local or global selection algorithms have been suggested, however they often fail to guarantee monotonicity.

2.2 The Monotonicity Conditions

In [9], Fritsch and Carlson derived necessary and sufficient conditions for a piecewise cubic interpolant to be monotonic. These conditions have frequently been used to develop a family of algorithms for global or local monotonic cubic interpolation. Let $\Delta y_i = y_{i+1} - y_i$ be the slope of the line segment connecting the data points to be interpolated. A curve is monotone on an interval if and only if there is no sign change in the derivative values in the interval. Hence, it is clear that the following is a necessary condition for monotonicity:

$$\begin{cases} sgn(y'_0) = sgn(y'_1) \\ \qquad\qquad = sgn(\Delta y_0), & \text{if } \Delta y_0 \neq 0, \\ y'_0 = y'_1 = 0, & \text{otherwise.} \end{cases} \quad (1)$$

Let $\alpha = \frac{y'_0}{\Delta y_0}$ and $\beta = \frac{y'_1}{\Delta y_0}$ (Without loss of generality, we assume that $\Delta y_0 \neq 0$ in the remainder of this section). Then, α and β are nonnegative if the requirement in Eq. (1) is satisfied. Now, the monotonicity conditions for the cubic Hermite interpolant $y = y(x)$ can be compactly expressed as follows (see Figure 3):



(a) The entire region (b) The region dissected

Figure 3: The monotonicity region [9]. The monotonicity of the spline from the cubic Hermite interpolation is guaranteed if the point $(\alpha, \beta) = (\frac{y_0'}{\Delta y_0}, \frac{y_1'}{\Delta y_0})$ resides in this region.

1. If $\alpha + \beta - 2 \leq 0$, then $y = y(x)$ is monotone on $[0, 1]$ if and only if the condition in Eq. (1) is satisfied.

2. If $\alpha + \beta - 2 > 0$, and the condition in Eq. (1) is satisfied, then $y = y(x)$ is monotone on $[0, 1]$ if and only if one of the following conditions is satisfied:

   a) $2\alpha + \beta - 3 \leq 0$,

   b) $\alpha + 2\beta - 3 \leq 0$, or

   c) $\phi(\alpha, \beta) \equiv \alpha - \frac{1}{3}\frac{(2\alpha+\beta-3)^2}{(\alpha+\beta-2)} \geq 0$.

Previous works on local interpolation have attempted to design monotone splines by choosing appropriate derivatives from neighboring slopes such that the monotonicity conditions are satisfied (for example, refer to [14, 15, 16]). Once the derivatives are fixed, however, a cubic interpolant in functional form is uniquely determined, in which case it is not possible to control the general appearance of splines. Notice that the Catmull–Rom spline is not effective for monotonicity because its derivatives often result in non-monotone splines. In the next section, we propose to use parametric cubic interpolation, based on the Catmull–Rom spline, that provides more flexibility in interpolation.

3 The New Controllable Local Monotonic Cubic Interpolation

3.1 Assigning the Derivatives in Catmull–Rom Fashion

In Catmull–Rom splines, the derivatives are set to the arithmetic mean of the two incident increments. Combined with the necessary condition in Eq. (1), the derivatives y_0' and y_1' are chosen in Catmull–Rom fashion as follows:

$$y_i' = \begin{cases} \frac{\Delta y_{i-1} + \Delta y_i}{2}, & \text{if } \Delta y_{i-1} \cdot \Delta y_i > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

The selected derivatives are the same as those used by the standard Catmull–Rom spline if the given four values y_{-1}, y_0, y_1, and y_2 are monotonic.

3.2 An Efficient Test of the Monotonicity Conditions

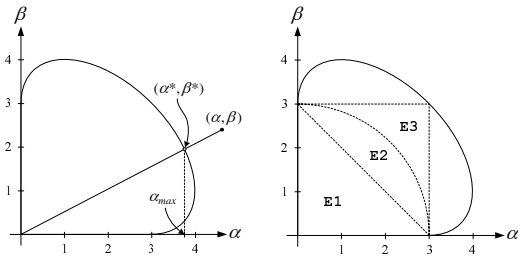
Once the derivatives at the two endpoints are determined, they must be checked to ensure they are *acceptable*. It is important to find efficiently whether a given pair $(\alpha, \beta) = (\frac{y_0'}{\Delta y_0}, \frac{y_1'}{\Delta y_0})$ satisfies the monotonicity conditions. Consider the line joining the origin and the point (α, β) (see Figure 4(a)). It can be shown that the line and the ellipse $\phi(\alpha, \beta) = 0$ intersect at $\alpha_{min} = \frac{3}{1+\sqrt{m}+m}$ and $\alpha_{max} = \frac{3}{1-\sqrt{m}+m}$, where $m = \frac{\beta}{\alpha}$ for $\alpha \neq 0$. Therefore, we can say that the pair (α, β) satisfies the monotonicity constraints if

$$\begin{cases} \alpha \leq \frac{3}{1-\sqrt{m}+m}, & \text{if } \alpha \neq 0 \\ \beta \leq 3, & \text{otherwise.} \end{cases}$$

The inequality $\alpha \leq \frac{3}{1-\sqrt{m}+m}$ can be written equivalently, and somewhat more efficiently, as $\alpha+\beta-3 \leq \sqrt{\alpha\beta}$. When our cubic interpolation scheme is coded, we use an even more efficient expression, which avoids potentially expensive square-root operations as follows:

$$\alpha + \beta - 3 \leq 0 \textbf{ or } \{\textbf{not}(\alpha + \beta - 3 \leq 0) \textbf{ and } (\alpha + \beta - 3)^2 \leq \alpha\beta\}. \quad (3)$$

In fact, it is possibly more efficient to replace the logical expression $\alpha + \beta - 3 \leq 0$ by ($\alpha \leq 3$ **and** $\beta \leq 3$). Notice that these



(a) The test of monotonicity constraints

(b) The approximate subregions

Figure 4: The monotonicity region revisited. When the pair (α, β) is outside the region, it is moved into it along the line connecting the pair with the origin to satisfy the monotonicity constraints.

two expressions correspond to the triangular region (E1) and the square region (E3), respectively (see Figure 4(b)), where the latter covers a larger area. When the data to be interpolated are from smooth parts of functions, as is often the case, the selection of derivatives in Catmull–Rom fashion most probably falls in the square region E3. Therefore, the monotonicity constraints can be tested cheaply in most cases.

3.3 Modification of the (α, β) Pair

When a pair (α, β) does not satisfy the monotonicity conditions, it must be modified appropriately. An effective way is to move the pair along the line connecting the pair with the origin to where the line intersects with the ellipse (see Figure 4(a) again). In this case, the new pair (α^*, β^*) is expressed as follows:

$$\alpha^* = \frac{3}{1 - \sqrt{m} + m}, \ \beta^* = m \cdot \alpha^*, \quad \text{where } m = \frac{\beta}{\alpha} \text{ for } \alpha \neq 0. \tag{4}$$

Of course, if α is zero, the pair is chosen to be $(\alpha^*, \beta^*) = (0, 3)$ only when $\beta > 3$.

Other selection methods are possible by putting the new pair on the boundary of or inside the approximate subregion E1, E3, or E2, a circle centered at the origin of radius 3. While it is probably cheaper to compute, only a part of the monotonicity region is utilized (even E3 covers only 67.9% of the entire region). Rather than clamping the acceptable pair to a restricted subregion, our interpolation scheme fully utilizes the monotonicity region. Observe that an (α, β) pair closer to the origin results in a "tighter" spline segment. Because the new pair (α^*, β^*) is on the boundary of the region, it corresponds to a spline that just meets the monotonicity constraints, possibly containing an inflection point. Its shape can be made flatter by moving it further towards the origin. The global shape control will be discussed later.

3.4 Interpolation with Local Tension Control

When the initially chosen derivatives y'_0 and y'_1 are found to satisfy the monotonicity conditions, the cubic Hermite interpolation in functional form can be used. However, if this is not the case, the slopes must be modified properly to meet them. The functional cubic curve with only four degrees of freedom is not flexible enough to satisfy the monotonicity constraints without changing the slopes. Because we intend to design a local scheme that basically uses a four-point stencil, such slope modification makes it inefficient to obtain at least G^1 continuous splines. Of course, fixed slopes that guarantee the monotonicity could be used, as mentioned before. However, they are not well-suited either, because we want to develop a controllable interpolation scheme. Instead of the functional interpolation, we exploit the parametric cubic interpolation that still has two degrees of freedom after the Hermite interpolation. In this approach, monotonicity is achieved by controlling the tensions at the endpoints while maintaining the slope directions. As will be explained, the extra degrees of freedom allow a useful controllability in interpolation.

Let $P_i = (i, y_i)$ and $T_i = (1, y'_i)$ $(i = 0, 1)$ be two endpoints and two tangent vectors, respectively, of a curve to be interpolated, where the slopes y'_0 and y'_1 are the initial derivatives assigned as in Eq. (2) (see Figure 5). Then the standard parametric cubic Hermite interpolation $C = C(t)$ is expressed as follows: $C(t) = (x(t), y(t)) = (2t^3 - 3t^2 + 1)P_0 + (-2t^3 + 3t^2)P_1 + (t^3 - 2t^2 + t)T_0 + (t^3 - t^2)T_1 \ \ (0 \leq t \leq 1)$.

With the introduction of *local* tension parameters τ_0 and τ_1 at the interpolation points, the two tangent vectors can be written as $T_i =$

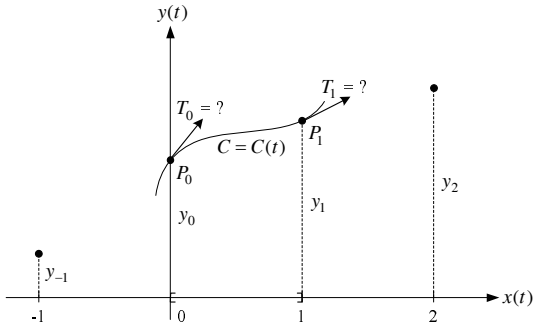


Figure 5: Cubic Hermite interpolation in parametric form. Parametric curves offer more degrees of freedom in controlling their geometric properties.

$(1-\tau_i, (1-\tau_i)\cdot y_i')$ with default value $\tau_i = 0$. Increasing the tension parameter τ_i from 0 to 1 reduces the magnitude of the tangent vector, resulting in a tighter curve near the endpoint P_i. Because the interpolation curve loops at the endpoint for $\tau_i > 1$, we only consider tension values less than one.

Notice that an interpolation curve $C(t) = (x(t), y(t))$ is monotonic on $[0,1]$ if both $x = x(t)$ and $y = y(t)$ are monotone in the interval. When the local tension parameters τ_0 and τ_1, initially set at zero, result in a non-monotone spline, the key is to adjust them so that both functions become monotonic. We flatten non-monotone curve segments, through the tension control, to enforce the monotonicity constraints. Therefore, it is reasonable to assume that the local tension parameters range in the interval $[0,1)$. Then, the function $x = x(t)$ is always monotone on $[0,1]$ because its (α, β) pair with $\alpha = x_0' = 1-\tau_0$ and $\beta = x_1' = 1-\tau_1$ always meets the first constraint of the monotonicity conditions ($\alpha + \beta - 2 \leq 0$). For the function $y = y(t)$, its (α, β) pair is checked to see if the initial derivatives result in a monotone function. If not, the adjusted tensions τ_0 and τ_1 are computed through the new pair (α^*, β^*) on the boundary of the monotonicity region as follows: $1-\tau_0 = \frac{\alpha^*}{\alpha}$ and $1-\tau_1 = \frac{\beta^*}{\beta}$.

A drawback with this parametric interpolation scheme is that x is not an independent variable as in the functional case. Given $x^* \in [0,1]$, the interpolation must be carried out via the parameter t by first finding $t^* = x^{-1}(x^*)$ and then evaluating $y^* = y(t^*)$. Because the function $x = x(t)$ is monotone on $[0,1]$, a unique t^* exists. The inversion can be performed by a few Newton–Raphson iterations on $f(t) = x(t) - x^* = 0$, which converges rapidly and requires little computation. Usually, we find that four to five iterations are good enough.

3.5 Introducing the Global Tension Parameter

In the interpolation scheme described here, the Catmull–Rom spline is employed if the generated curve segment is monotone. If this is not the case, we choose a spline that just meets the monotonicity conditions. Sometimes, it is highly desirable to control the general behavior of the interpolation by adjusting the shapes of the splines. Unlike the functional interpolation, the parametric cubic interpolation is flexible enough to allow the user to select a proper interpolant from a space of splines that share the same tangential lines at the interpolation points.

In our technique, a parameter τ_{global} ($0 \leq \tau_{global} \leq 1$), called a *global* tension, is employed as an interpolation control. Instead of the tangent vector $T_i = (1-\tau_i, (1-\tau_i)\cdot y_i')$, we use $T_i = ((1-\tau_{global})\cdot(1-\tau_i), (1-\tau_{global})\cdot(1-\tau_i)\cdot y_i')$ in computing the parametric Hermite curve $C(t)$. When τ_{global} increases from zero to one, the interpolant becomes flattened, ending up in linear interpolation at $\tau_{global} = 1$ (The new local monotonic cubic interpolation algorithm is summarized in Figure 6). Figure 7 illustrates how the monotonic interpolation spline in Figure 1(b) changes gradually as the global tension parameter increases. We observe a continuous spectrum of cubic to linear interpolants parameterized by τ_{global}, thus enabling the user to control the interpolation.

While we have described one technique for interpolation control, it should be mentioned that other control strategies are also possible. For instance, we could move the (α, β) pair as closely as possible to the boundary of the monotonicity region wherever it is located in the α-β plane. If the pair is outside the region, it is moved to the boundary as explained before. When the pair moves towards the boundary from inside, we must take care not to violate the monotonicity conditions for the function $x = x(t)$, moving it up to the boundary or the critical point whichever comes first. The new pair found

Input: y_{-1}, y_0, y_1, y_2, x^*
Output: $y^* = \mathsf{CUBIC\_INTERP}(y_{-1}, y_0, y_1, y_2, x^*; \tau_{global})$
1. Compute the slopes to y'_0 and y'_1 as in Eq. (2).
2. Check if the monotonicity conditions are met
   for $(\alpha, \beta) = (\frac{y'_0}{\Delta y_0}, \frac{y'_1}{\Delta y_0})$.
3. **if** they are not satisfied **then**
4. adjust the local tensions via $1 - \tau_0 = \frac{\alpha^*}{\alpha}$ and $1 - \tau_1 = \frac{\beta^*}{\beta}$.
5. Let $\sigma_0 = (1 - \tau_{global}) \cdot (1 - \tau_0)$ and $\sigma_1 = (1 - \tau_{global}) \cdot (1 - \tau_1)$.
6. Find t^* such that $t^* = x^{-1}_{[0,1,\sigma_0,\sigma_1]}(x^*)$.
7. Evaluate $y^* = y_{[y_0, y_1, \sigma_0 \cdot y'_0, \sigma_1 \cdot y'_1]}(t^*)$.

Figure 6: The controllable local monotonic cubic interpolation algorithm.

in this way produces one of the "slackest" monotonic splines that are possible with the assigned derivatives. The resulting spline may look too loose, as shown in Figure 8. However, we observe that the global tension control offers another nice spectrum of monotonic interpolants, smoother at the same τ_{global}.

Before finishing this description, we briefly mention the geometric continuity achieved through the new controllable interpolation scheme. It is clear that the generated splines meet each other with G^1 continuity because the local tension parameter only changes the magnitude of the tangent vector. It is possible to make them C^1 continuous using a six-point stencil $\{y_{-2}, y_{-1}, \cdots, y_3\}$ instead of the four-point stencil $\{y_{-1}, \cdots, y_2\}$. The local tension parameter τ_i for each interpolation point x_i $(i = 0, 1)$ is set to the maximum of the two local tension parameters τ_i^L and τ_i^R that guarantee the monotonicity on the two incident intervals $[i-1, i]$ and $[i, i+1]$, respectively. This modification would produce C^1 continuous splines, although it is not evident that the additionally required computational cost is worth paying.

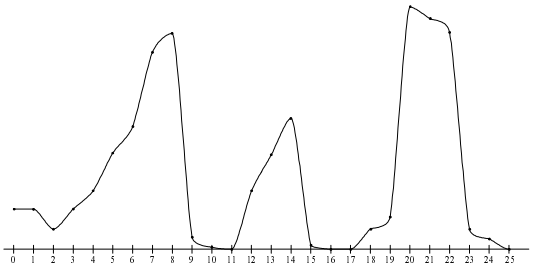
4 Applications to Fluid Simulations

To understand how our interpolation filter affects fluid animations, we have added our controllable interpolation scheme to our fluid dynamics solver, which is based on the computational models presented in [18, 7]. When the Navier-Stokes equations are integrated over time in three-dimensional space, a vector field for the velocity and two scalar fields for the density and the temperature should be resampled through interpolation in the advection stage.

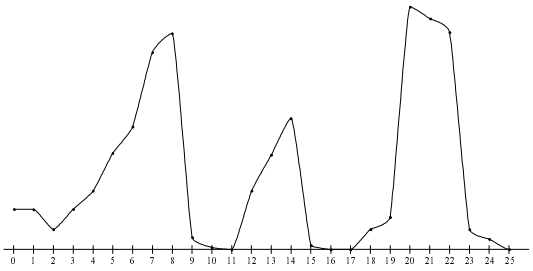
4.1 Effects of the Controllable Cubic Interpolation on Visual Appearance

In an attempt to create smoke animations with different appearances, we have experimented with the controllable cubic interpolation filter in several ways. In the first example, a set of different global tension parameters τ_{global} was tested to resample both the density and the temperature, while the velocity was resampled using a linear interpolant only. All the other animation and rendering parameters were identical. Figure 9 demonstrates five animation sequences of two rising-and-reacting smokes, where the top row was generated with $\tau_{global} = 0.0$, and subsequent rows were generated using $0.25, 0.5, 0.75$, and 1.0, respectively. The first row corresponds to a *full* cubic monotonic interpolant, while the last row with $\tau_{global} = 1.0$ corresponds to a linear interpolant. As also discussed in [7], we find that the cubic interpolant produces a very abundant smoke simulation with fine detail.

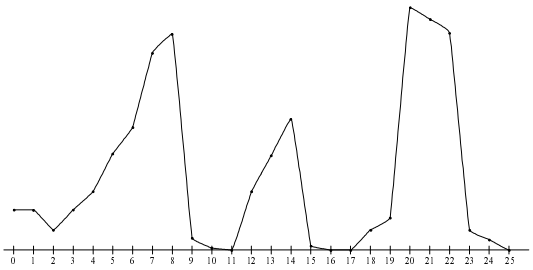
On the other hand, the linear interpolant generates a light simulation in which the smoke dissipates too quickly because of its limited numerical accuracy. As the global tension parameter varies from 0.0 to 1.0 (top to bottom), we obtain a spectrum of animations with continuously changing appearance. The generated smoke tends to be intense and swirling when τ_{global} is close to 0.0 but becomes rather smooth and dissipated as τ_{global} approaches 1.0. Notice that the animation results are correlated very naturally with the interpolation curves shown in



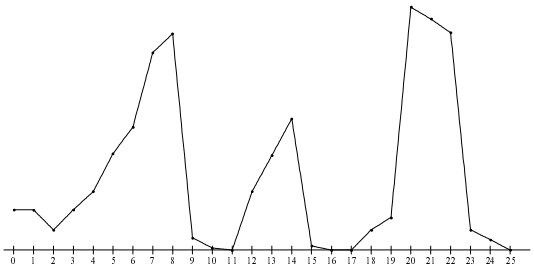
(a) $\tau_{global} = 0.25$



(b) $\tau_{global} = 0.5$



(c) $\tau_{global} = 0.75$



(d) $\tau_{global} = 1.0$

Figure 7: A spectrum of controllable monotonic cubic interpolants. As the global tension parameter τ_{global} increases, the monotonic spline in Figure 1(b) changes continuously to a linear interpolant.

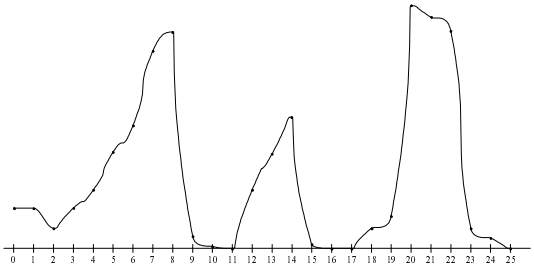


Figure 8: Another strategy for interpolation control. A very slack spline curve with $\tau_{global} = 0.0$ is obtained, where each spline segment has been pushed to the limit.

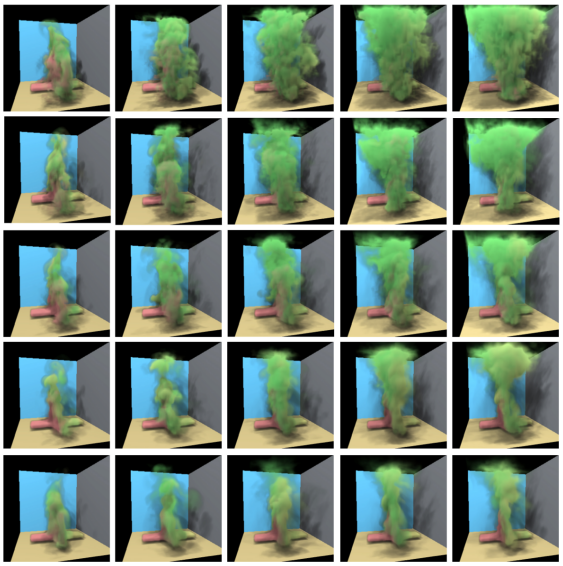


Figure 9: Comparison of rising-and-reacting smokes created with our local monotonic cubic interpolant. The new interpolation scheme was applied to the evolving smoke's density and temperature with different global tension parameters $\tau_{global} = 0.0, 0.25, 0.5, 0.75$, and 1.0 (top to bottom), while a linear interpolant was used to resample the velocity. As the global tension parameter varies from 0.0 to 1.0, we obtain a spectrum of smoke simulations with gradually changing behaviors.

Figure 1(b) and Figure 7.

We have also employed the controllable filter to interpolate the smoke's velocity as well as its density and temperature. The monotonic cubic interpolant with $\tau_{global} = 0.0$ was applied to the velocity vector in the example shown in the first row of Figure 9, and the same cubic interpolant was used for both the density and the temperature. All the other animation parameters were the same. The simulation result is given in the first row of Figure 10, where the same time frames are displayed for comparison. We find that more fine detail of smoke is produced as the cubic interpolant reduces the amount of numerical dissipation in the resampling of the velocity. As a side effect, we observe that the smoke does not rise as abundantly as in the example where a linear interpolant was employed for the velocity. It seems that with the added numerical detail, the vorticity confinement force that was appropriate in the previous example becomes excessive, hence dominating the buoyancy force.

We attempted to control the way that the smoke flows by lowering the coefficient ϵ of the vorticity confinement force $\mathbf{f}_{conf} = \epsilon\ h\ (\mathbf{N} \times \omega)$ (refer to [7] for detail). ϵ specifies the amount of small scale detail added back into the smoke field. When ϵ was set to 3.0 instead of 12.0, we were able to produce a smoke simulation with a general appearance somewhat similar to the animation in the first row of Figure 9, but it is filled with slightly more fine detail (see the animation frames in the second row of Figure 10). When ϵ was lowered to 0.0, that is, no vorticity confinement force was added, the dominating buoyancy force makes the smoke flow upwards aggressively. This effect is seen clearly in the animation sequence shown in the third row of Figure 10, where small scale detail of the smoke disappears because of the loss of the vorticity confinement force.

Lastly, we tried to control the smoke animation in a different way. Rather than decreasing the vorticity confinement force in the animation given in the first row ($\tau_{global} = 0.0$ and $\epsilon = 12$), we changed the global tension parameter of the monotonic cubic filter. The animation sequence in the fourth row of Figure 10 was created with a new $\tau_{global} = 0.5$ and the same $\epsilon = 12.0$, where the employed interpolation filter is somewhat in the middle of the cubic and linear interpolants.

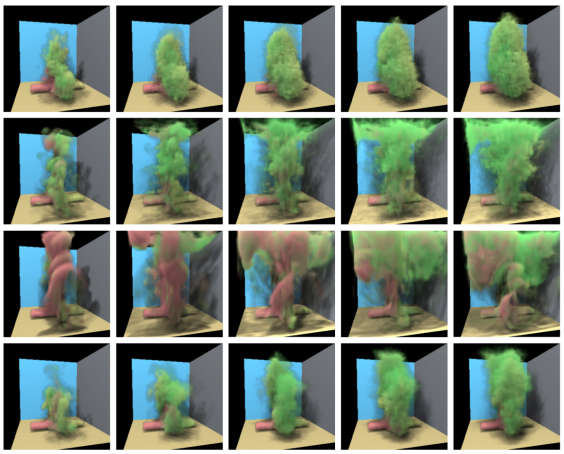


Figure 10: Comparison of rising-and-reacting smoke created by additionally applying our local monotonic cubic interpolant to the velocity resampling. As shown in the first row, more fine detail was possible by virtue of the increased numerical detail in the process of resampling the smoke's velocity. Compared to the first animation in Figure 9, it seems that the vorticity confinement force dominates the buoyancy force. In the first three rows, three different vorticity confinement coefficients $\epsilon = 12.0, 3.0,$ and 0.0 with the same $\tau_{global} = 0.0$ were tested to understand how the vorticity confinement force affects the behavior of smoke. The animation in the fourth row was created with $\epsilon = 12.0$, but with $\tau_{global} = 0.5$.

Curiously, the smoke's volume varies in similar fashion as in the first row. However, the smoke becomes, as expected, somewhat smoother and lighter because of the numerical dissipation introduced.

4.2 Computational Overheads of the Controllable Cubic Interpolation

Table 1 summarizes the timing performance of three different filters applied to the scenes in Figure 9 and 10: Linear for the tri-linear filter, Cubic [7] for the C^0-continuous monotonic tri-cubic filter used in [7], and CLM-Cubic for the controllable G^1-continuous monotonic tri-cubic filter. In the table, DT-Only denotes the case where the tri-cubic filters were applied to the

evolution of the density and temperature only, while the velocity were additionally resampled using the tri-cubic filters in DTV. The timings were measured on a PC, equipped with a 2.0 GHz Intel Xeon CPU and 1 GB RAM, and show the average times per time step, necessary for updating the velocity and evolving the density and temperature on a grid of resolution $60 \times 60 \times 60$.

(sec.)

	Linear	Cubic [7]	CML-Cubic
DT-Only	5.55	6.62	6.98
DTV		8.56	9.76

Table 1: Timing statistics. The figures compares the three filters in terms of the average timings per time frame, taken for updating the velocity and evolving the density and temperature on a $60 \times 60 \times 60$ grid. Two cases were tested where the cubic filters were first applied to the advection of the density and temperature, then the velocity was additionally resampled using the cubic filter in the second.

In this experiment, the respective filters were applied to resample the points computed by tracing the characteristics curves backward in the advection stage of the 3D fluid simulator, in which, for the cubic filters, each one-dimensional filter is repeatedly applied 21 times per resampling (16 times in the x direction, 4 times in the y direction, and finally once in the z direction). As expected, the experimental results indicate that the computing cost of the controllable filter is high compared to the other two filters. In particular, the cost of CLM-Cubic, based on parametric cubic interpolation, is a little bit higher than that of Cubic [7], based on functional cubic interpolation. The additional cost mainly comes from the Newton-Raphson iterations that are required to invert the cubic function $x = x(t)$. Since this monotone function behaves well on $[0, 1]$, only a few iterations on the simple cubic polynomial are enough, which is quite cheap computationally.

It should be mentioned that it is often difficult to tell the difference between the interpolation results obtained using Cubic [7] and CLM-Cubic with $\tau_{global} = 0.0$. Particularly when the simulated fluid is visualized, it is not easy to distinguish the difference between the rendered images. Notice that the little cost increase is the price that we must pay to make the resulting interpolants *monotone*, G^1*-continuous* and, above all, *controllable*. These mathematically fine properties for the cubic interpolation often lead to effective applications. For instance, the presented controllable interpolation scheme is quite convenient when the user may want to interactively choose an interpolation filter from a continuous spectrum of linear to cubic monotonic filters. Figure 11 shows a snapshot where the global tension parameter is being interactively adjusted in a 2D fluid simulator to see how the visual appearance of the fluid changes (see also the attached video). Here, we have mapped the controllable monotonic bi-cubic filter onto the pixel shader hardware of NVIDIA's GeForce 6800 GT GPU. While employing the controllable interpolation in the simulation, again, results in a lower frame rate, we find it quite useful to be able to control the filter's behavior interactively.

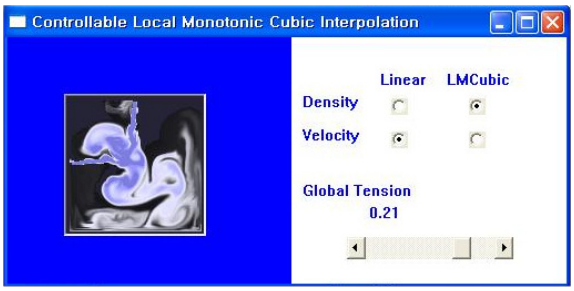


Figure 11: Interactive control of the global tension parameter. The presented cubic filter has been implemented using the pixel shader on a programmable GPU, which allows an interactive control of the global tension in the 2D fluid simulation.

5 Conclusions

The motivation of this paper was to understand how different interpolation filters employed in the simulation stage affect fluid animations. In achieving this goal, we have designed a local monotonic cubic interpolation scheme that allows the user to control its behavior. Through the use of the global tension parameter, the proposed interpolation technique is able to offer a

spectrum of cubic to linear interpolants, filling the gap between them.

The experiments with our smoke simulation solver indicate that fluid animations are greatly influenced by the choice of interpolation filter that is applied to resampling data such as the velocity, the density, and the temperature. Conversely, this fact implies that the controllable interpolation filter may be used in modeling fluid animation creatively, although this approach is not necessarily physically based. We have given some examples where a wide range of smoke effects were developed through intuitive control of the interpolation filter.

Acknowledgements

This research was supported by the Ministry of Information and Communication of Korea under the Information Technology Research Center support program.

References

[1] D. Mitchell and A. Netravali. Reconstruction filters in computer graphics. In *Proc. of ACM SIGGRAPH 1988*, pages 221–228, 1988.

[2] S. Marschner and R. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proc. of IEEE Visualization 1994*, pages 100–107, 1994.

[3] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. Evaluation and design of filters using a Taylor series expansion. *IEEE Trans. Visualization and Computer Graphics*, pages 184–199, 1997.

[4] T. Möller, K. Mueller, Y. Kurzion, R. Machiraju, and R. Yagel. Design of accurate and smooth filters for function and derivative reconstruction. In *Proc. of IEEE Symposium on Volume Visualization*, pages 143–151, 1998.

[5] T. Theußl, H. Hauser, and E. Gröller. Mastering windows: Improving reconstruction. In *Proc. of IEEE Symposium on Volume Visualization*, pages 101–108, 2000.

[6] M. Hadwiger, T. Theußl, H. Hauser, and E. Gröller. Hardware-accelerated high-quality filtering on PC hardware. In *Proc. of Vision, Modeling, and Visualization 2001*, pages 105–112, 2001.

[7] R. Fedkiw, J. Stam, and H. Jensen. Visual simulation of smoke. In *Proc. of ACM SIGGRAPH 2001*, pages 23–30, 2001.

[8] S. Conte and C. de Boor. *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw-Hill, Inc., third edition, 1980.

[9] F. Fritsch and R. Carlson. Monotone piecewise cubic interpolation. *SIAM J. Numerical Analysis*, 17(2):238–246, 1980.

[10] G. Wolberg and I. Alfy. Monotonic cubic spline interpolation. In *Proc. of Computer Graphics International 1999*, pages 188–195, 1999.

[11] D. Schweikert. An interpolation curve using a spline in tension. *J. Math. and Phys.*, 45:312–317, 1966.

[12] N. Sapidis, P. Kaklis, and T. Loukakis. A method for computing the tension parameters in convexity preserving spline-in-tension interpolation. *Numer. Math.*, 54:179–192, 1988.

[13] T. Foley. Interpolation with interval and point tension controls using cubic weighted ν-splines. *ACM Trans. Math. Software*, 13(1):68–96, 1987.

[14] J. Butland. A method of interpolating reasonable-shaped curves through any data. In *Proc. of Computer Graphics 80*, pages 409–422. Online Publications Ltd., 1980.

[15] F. Fritsch and J. Butland. An improved monotone piecewise cubic interpolation algorithm. Technical Report UCRL-85104, Lawrence Livermore Laboratory, Livermore, CA, October 1980.

[16] F. Fritsch and J. Butland. A method for constructing local monotone piecewise cubic interpolants. *SIAM J. Sci. Stat. Comput.*, 5(2):300–304, 1984.

[17] C. Manni. C^1 comonotone Hermite interpolation via parametric cubics. *J. Comput. Appl. Math.*, 69(1):143–157, 1996.

[18] J. Stam. Stable fluids. In *Proc. of ACM SIGGRAPH 1999*, pages 121–128, 1999.