AN ALGORITHM FOR CONSTRUCTING MONOTONE QUINTIC INTERPOLATING SPLINES

Thomas C. H. Lux
Layne T. Watson
Tyler H. Chang

Departments of Computer Science, Mathematics, and Aerospace and Ocean Engineering
Virginia Polytechnic Institute & State University
2000 Torgersen Hall, Blacksburg, VA, USA
{tchlux,ltw,thchang}@vt.edu

Li Xu
Yueyao Wang
Yili Hong

Department of Statistics
Virginia Polytechnic Institute & State University
213 Hutcheson Hall, Blacksburg, VA, USA
{lix1992,yueyao94,yilihong}@vt.edu

ABSTRACT

A novel algorithm for computing monotone order six piecewise polynomial interpolants is proposed. Algebraic constraints for enforcing monotonicity are provided that align with quintic monotonicity theory. The algorithm is implemented, tested, and applied to several sample problems to demonstrate the improved accuracy of monotone quintic spline interpolants compared to the previous state-of-the-art monotone cubic spline interpolants.

Keywords: monotone, quintic spline, Hermite interpolation, sixth order polynomial.

1 INTRODUCTION AND MOTIVATION

Many domains of science rely on smooth approximations to real-valued functions over a closed interval. Piecewise polynomial functions (splines) provide the smooth approximations for animation in graphics (Herman and Oftedal 2015, Quint 2003), aesthetic structural support in architecture (Brennan 2019), efficient aerodynamic surfaces in automotive and aerospace engineering (Brennan 2019), prolonged effective operation of electric motors (Berglund et al. 2009), and accurate nonparametric approximations in statistics (Knott 2012). While polynomial interpolants and regressors apply broadly, splines are often a good choice because they can approximate globally complex functions while minimizing the local complexity of an approximation.

It is often the case that the true underlying function or phenomenon being modeled has known properties e.g., convexity, positivity, various levels of continuity, or monotonicity. Given a reasonable amount of data, it quickly becomes difficult to achieve desirable properties in a single polynomial function. In general, the maintenance of function properties through interpolation/regression is referred to as *shape preserving* (Fritsch and Carlson 1980, Gregory 1985). The specific shapes this work will achieve in approximations are monotonicity and C^2 continuity. These properties are chiefly important to the approximation of cumulative distribution functions and subsequently the effective generation of random numbers from a specified distribution.

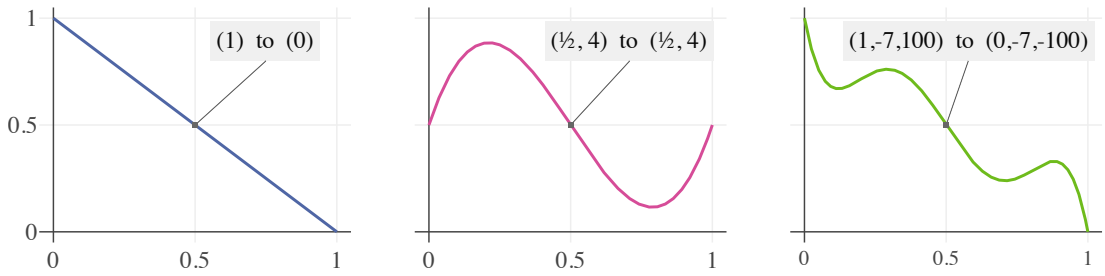

Figure 1: Example polynomials that interpolate function values at the ends of the interval $[0,1]$. The first only interpolates the function values $g_2(0) = 1$ and $g_2(1) = 0$, making it the order two polynomial $g_2(x) = 1 - x$. For the second plot $g_4(x) = 8x^3 - 12x^2 + 4x + 1/2$, which is order four and interpolates the values $g_4(0) = 1/2$, $g_4'(0) = 4$, $g_4(1) = 1/2$, $g_4'(1) = 4$. Finally the third plot shows the order six polynomial $g_6(x) = -64x^5 + 160x^4 - 140x^3 + 50x^2 - 7x + 1$ interpolating the function values $g_6(0) = 1$, $g_6'(0) = -7$, $g_6''(0) = 100$, $g_6(1) = 0$, $g_6'(1) = -7$, $g_6''(1) = -100$. Notice that interpolating the same fixed number of function values at each endpoint will always result in an even order interpolating polynomial.

In statistics especially, the construction of a monotone interpolating spline that is C^2 continuous is meaningfully useful (Ramsay et al. 1988). A function with these properties could approximate a cumulative distribution function to a high level of accuracy with relatively few intervals. A twice continuously differentiable approximation to a cumulative distribution function (CDF) would also produce a corresponding probability density function (PDF) that is continuously differentiable, which is a property many standard parametric distributions maintain.

The currently available software for monotone piecewise polynomial interpolation includes quadratic (He and Shi 1998), cubic (Fritsch and Carlson 1980), and (with limited application) quartic (Wang and Tan 2004, Piah and Unsworth 2011, Yao and Nelson 2018) cases. In addition, a statistical method for bootstrapping the construction of an arbitrarily smooth monotone fit exists (Leitenstorfer and Tutz 2006), but the method does not take advantage of known analytic properties related to quintic polynomials. Theory has been provided for the quintic case (Ulrich and Watson 1994, Hess and Schmidt 1994), however this theory has not yet been used to construct a monotone quintic spline interpolation routine. Recent work suggests that the lack of quintic software may be due to a general unawareness of the theory (Xie et al. 2018).

The importance of piecewise quintic interpolation over lower order approximations can be simply demonstrated. In general, the order of a polynomial determines the number of function values it can interpolate, and the growth rate of error away from the interpolated function values. As demonstrated in Figure 1, it can be seen that matching a value at either end of the interval requires an order two (linear) approximation and each additional derivative at the ends of the interval raises the necessary polynomial order by two. Hence, only even order (odd degree) interpolating splines are broadly applicable. The body of this work is composed of a novel algorithm for enforcing monotonicity on quintic polynomial pieces, then extending that solution to work on quintic splines.

The major contribution of this work is an algorithm for constructing monotone quintic interpolating splines that utilizes existing quintic monotonicity theory. The remainder of this paper is structured as follows: Section 1.1 summarizes the existing monotone cubic spline interpolation methodology, Section 2 presents an algorithm for constructing monotone quintic spline interpolants, Section 3 offers experiments and results with cubic and quintic methods, Sections 4 and 5 discuss results and conclude.

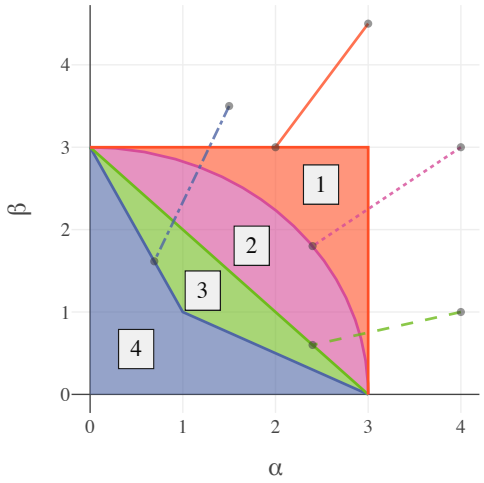

Figure 2: These are the feasible regions of monotonicity for cubic splines and the projections that make a cubic polynomial piece monotone. The regions themselves are numbered 1–4 corresponding to their original description in Fritsch and Carlson (1980). One point is projected onto each region as a demonstration.

1.1 Computing a Monotone Cubic Interpolant

The current state-of-the-art monotone interpolating spline with a mathematical software implementation is piecewise cubic, continuously differentiable, and was first proposed in (Fritsch and Carlson 1980) then expanded upon in (Carlson and Fritsch 1985). Let $\pi : x_0 = k_1 < k_2 < \cdots < k_n = x_1$ be a partition of the interval $[x_0, x_1]$. Let $f : \mathbb{R} \to \mathbb{R}$ be C^2, and $\{f(k_i)\}_{i=1}^n$ and $\{\Delta_i\}_{i=1}^n$ be given sets of function and derivative values at the partition points for a monotone function f. Either $f(k_i) \leq f(k_{i+1})$, $i = 1, \ldots, n-1$, and $\Delta_i \geq 0$, $i = 1, \ldots, n$, or $f(k_i) \geq f(k_{i+1})$, $i = 1, \ldots, n-1$, and $\Delta_i \leq 0$, $i = 1, \ldots, n$. Let $\hat{f}$ be a piecewise cubic polynomial defined in each subinterval $I_i = [k_i, k_{i+1}]$ by

$$
\begin{aligned}
h_i &= k_{i+1} - k_i, \quad u(t) = 3t^2 - 2t^3, \quad p(t) = t^3 - t^2, \\
\hat{f}(x) &= f(k_i)u\big((k_{i+1} - x)/h_i\big) + f(k_{i+1})u\big((x - k_i)/h_i\big) \\
&\quad - h_i\Delta_i p\big((k_{i+1} - x)/h_i\big) + h_i\Delta_{i+1} p\big((x - k_i)/h_i\big).
\end{aligned}
$$

Notice that a trivially monotone spline results when $\Delta_i = 0$, for $i = 1, \ldots, n$. However, such a spline has too many *wiggles* for most applications. Carlson and Fritsch (1985) show that simple conditions on the derivative values can guarantee monotonicity, and that these conditions can be enforced in a way that ensures modifications on one interval will not break the monotonicity of cubic polynomials over any neighboring intervals. If $f(k_i) = f(k_{i+1})$, take $\Delta_i = \Delta_{i+1} = 0$ and $\alpha = \beta = 1$, otherwise let $\alpha = \big(\Delta_i(k_{i+1} - k_i)\big)/\big(f(k_{i+1}) - f(k_i)\big)$ and $\beta = \big(\Delta_{i+1}(k_{i+1} - k_i)\big)/\big(f(k_{i+1}) - f(k_i)\big)$. Monotonicity of a cubic polynomial over a subinterval can be maintained by ensuring that α and β reside in any of the regions depicted in Figure 2.

The actual region of monotonicity for a cubic polynomial is larger, but projection of (α, β) into one of these regions ensures that monotonicity will be achieved and not violated for neighboring regions. The user must decide which region is most appropriate for the projections based on the application, Fritsch and Carlson recommend using Region 2.

While the cubic monotonicity case affords such a concise solution, the region of monotonicity is not so simple in the quintic case. In the next section, an algorithm for performing a projection similar to those for cubic polynomials is proposed.

2 MONOTONE QUINTIC INTERPOLATION

The following section is composed of three algorithms that together are used to construct a monotone quintic spline interpolant. Without loss of generality, the algorithms will only consider the monotone increasing (nondecreasing) case. The monotone decreasing case is handled similarly. Algorithm 1 checks monotonicity, Algorithm 2 enforces monotonicity on an order six polynomial piece of the quintic spline, and Algorithm 3 uses the previous two algorithms to enforce monotonicity for the entire quintic spline.

In pseudocode, a function `binary_search(`g, a, b`)` is used, where $a, b \in S \subset \mathbb{R}^p$ for convex S, $g : S \to \{\texttt{FALSE}, \texttt{TRUE}\}$ is a right continuous Boolean function, $g(b) = \texttt{TRUE}$, and for $\mu \in [0,1]$, $g\big((1-\mu)a + \mu b\big) = \texttt{TRUE} \implies g\big((1-\nu)a + \nu b\big) = \texttt{TRUE}$ for $\mu \leq \nu \leq 1$. The search returns $\big((1-c)a + cb\big)$ for the smallest $c \in [0,1]$ such that $g\big((1-c)a + cb\big) = \texttt{TRUE}$.

These algorithms make use of first and second derivatives of the approximated function where the derivative operator is denoted with D. In the case that the first and second derivative information is not provided along with function values, the derivatives are estimated with finite differences of the function values. The final quintic spline is represented as a piecewise polynomial using the Newton form for each polynomial piece, and ultimately converted to a B-spline representation for evaluation.

Algorithm 1: `is_monotone`$\big(x_0, x_1, f\big)$

where x_0, $x_1 \in \mathbb{R}$, $x_0 < x_1$, and f is an order six polynomial defined by $f(x_0)$, $Df(x_0)$, $D^2 f(x_0)$, $f(x_1)$, $Df(x_1)$, $D^2 f(x_1)$. Returns `TRUE` if f is monotone increasing on $[x_0, x_1]$.

1: `if` $\big(f(x_0) = f(x_1)\big)$ `and not` $\big(0 = Df(x_0) = Df(x_1) = D^2 f(x_0) = D^2 f(x_1)\big)$
   `return FALSE`
   `end if`
2: `if` $\big(Df(x_0) < 0$ `or` $Df(x_1) < 0\big)$
   `return FALSE`
   `end if`

The necessity of these first two steps follows directly from the fact that f is C^2. The next case is in accordance with a simplified condition for quintic monotonicity that reduces to one of cubic positivity studied in Schmidt and Heß (1988), where α, β, γ, and δ are defined in terms of values and derivatives of f at x_0 and x_1. Step 5 checks for the necessary condition that $\alpha \geq 0$, Step 6 checks $\beta \geq \alpha$, and Step 7 checks $\gamma \geq \delta$, all from Schmidt and Heß (1988). If all necessary conditions are met, then the order six piece is monotone and Step 8 concludes this check.

3: `if` $\big(Df(x_0) = 0$ `or` $Df(x_1) = 0\big)$
4: $w := x_0 - x_1$
   $v := f(x_0) - f(x_1)$
5: `if` $\big(D^2 f(x_1) > -4Df(x_1)/w\big)$ `return FALSE`
6: `if` $\big(D^2 f(x_1) < (3wD^2 f(x_0) - 24Df(x_0) - 32Df(x_1) + 60v/w)/(5w)\big)$ `return FALSE`
7: `if` $\big(D^2 f(x_0) < 3Df(x_0)/w\big)$ `return FALSE`
8: `return TRUE`
   `end if`

The following code considers the remaining case where $Df(x_0) \neq 0$ and $Df(x_1) \neq 0$.

9: $A := Df(x_0)\dfrac{x_1 - x_0}{f(x_1) - f(x_0)}$
   $B := Df(x_1)\dfrac{x_1 - x_0}{f(x_1) - f(x_0)}$

The variables A and B correspond directly to the theoretical foundation for positive quartic polynomials established in Ulrich and Watson (1994), first defined after Equation (18).

10: $\gamma_0 := 4\dfrac{Df(x_0)}{Df(x_1)}(B/A)^{3/4}$

$\gamma_1 := \dfrac{x_1 - x_0}{Df(x_1)}(B/A)^{3/4}$

$\alpha_0 := 4(B/A)^{1/4}$

$\alpha_1 := -\dfrac{x_1 - x_0}{Df(x_1)}(B/A)^{1/4}$

$\beta_0 := 30 - \dfrac{12\big(Df(x_0) + Df(x_1)\big)(x_1 - x_0)}{\big(f(x_1) - f(x_0)\big)\sqrt{A}\sqrt{B}}$

$\beta_1 := \dfrac{-3(x_1 - x_0)^2}{2\big(f(x_1) - f(x_0)\big)\sqrt{A}\sqrt{B}}$

The γ, α, and β terms with subscripts 0 and 1 are algebraic reductions of the simplified conditions for satisfying Theorem 2 in Equation (16) of Ulrich and Watson (1994). These terms with subscripts 0 and 1 make the computation of α, β, and γ functions of the second derivative terms, as seen in Step 11 below.

11: $\gamma := \gamma_0 + \gamma_1 D^2 f(x_0)$

$\alpha := \alpha_0 + \alpha_1 D^2 f(x_1)$

$\beta := \beta_0 + \beta_1\big(D^2 f(x_0) - D^2 f(x_1)\big)$

12: `if` $(\beta \le 6)$ `return` $\big(\alpha > -(\beta+2)/2\big)$

`else` `return` $\big(\gamma > -2\sqrt{\beta - 2}\big)$

`end if`

The reason for structuring the α, β, and γ computations in terms of the second derivative of f (seen in Step 11 of Algorithm 1) will become more apparent later. The next problem to consider is that of making a nonmonotone order six polynomial piece into a monotone one by modifying its first and second derivative values at the ends of an interval. Note that the actual value of the function at the ends of the interval is not modified, as the resulting polynomial needs to interpolate.

Algorithm 2: `make_monotone`$\big(x_0, x_1, f\big)$
where $x_0,\ x_1 \in \mathbb{R}$, $x_0 < x_1$, and f is an order six polynomial defined by $f(x_0)$, $Df(x_0)$, $D^2 f(x_0)$, $f(x_1)$, $Df(x_1)$, $D^2 f(x_1)$. Returns f monotone on $[x_0, x_1]$.

1: `if` $\big(f(x_1) = f(x_0)\big)$

$Df(x_0) := Df(x_1) := D^2 f(x_0) := D^2 f(x_1) := 0$

`return`

`end if`

2: $Df(x_0) :=$ `median`$\left(0,\ \ Df(x_0),\ \ 14\dfrac{f(x_1) - f(x_0)}{x_1 - x_0}\right)$

$Df(x_1) :=$ `median`$\left(0,\ \ Df(x_1),\ \ 14\dfrac{f(x_1) - f(x_0)}{x_1 - x_0}\right)$

The selection of values in Step 2 for $Df(x_0)$ and $Df(x_1)$ is suggested by Ulrich and Watson (1994) and also by Huynh (1993). These rules quickly enforce upper and lower bounds on derivative values to ensure quintic monotonicity is obtainable.

The next case that will be covered is in accordance with a simplified condition for quintic monotonicity that reduces to one of cubic positivity studied in Schmidt and Heß (1988), where α, β, γ, and δ are defined in terms of values and derivatives of f at x_0 and x_1. Steps 5 and 6 ensure that the following bounds on $D^2f(x_0)$ are compatible while Step 7 ensures that the conditions on $D^2f(x_1)$ are compatible. To conclude the simplified case, Step 8 first enforces $\gamma \geq \delta$, then Steps 9 and 10 enforce $\alpha \geq 0$ and $\beta \geq \alpha$ respectively. Combined, these three conditions derived from Equation (2.18) in Schmidt and Heß (1988) guarantee monotonicity of this polynomial piece.

3: `if` $\big(Df(x_0) = 0$ `or` $Df(x_1) = 0\big)$

4: $w := x_0 - x_1$

$v := f(x_0) - f(x_1)$

5: `if` $\big(5Df(x_0) + 4Df(x_1) > 20v/w\big)$

6: $Df(x_0) := Df(x_0)\ \ \max\left(0, \dfrac{20v}{w\big(5Df(x_0) + 4Df(x_1)\big)}\right)$

$Df(x_1) := Df(x_1)\ \ \max\left(0, \dfrac{20v}{w\big(5Df(x_0) + 4Df(x_1)\big)}\right)$

`end if`

7: $D^2f(x_0) :=\ \min\left(D^2f(x_0), \dfrac{4(2Df(x_0) + Df(x_1)) + 20v/w}{w}\right)$

8: $D^2f(x_0) :=\ \max\big(D^2f(x_0), 3Df(x_0)/w\big)$

9: $D^2f(x_1) :=\ \min\big(D^2f(x_1), -4Df(x_1)/w\big)$

10: $D^2f(x_1) :=\ \max\left(D^2f(x_1), \dfrac{3wD^2f(x_0) - 24Df(x_0) - 32Df(x_1) + 60v/w}{5w}\right)$

`return`

`end if`

The following code considers the case where $Df(x_0) \neq 0$ and $Df(x_1) \neq 0$.

11: $A := Df(x_0)\dfrac{x_1 - x_0}{f(x_1) - f(x_0)}$

$B := Df(x_1)\dfrac{x_1 - x_0}{f(x_1) - f(x_0)}$

12: `if` $\big(\max(A,B) > 6\big)$

$Df(x_0) := 6Df(x_0)\big/\max(A,B)$

$Df(x_1) := 6Df(x_1)\big/\max(A,B)$

`end if`

This ensures that (A,B) remains within a viable region of monotonicity (satisfying Theorem 4, seen in Fig. 6 of Ulrich and Watson (1994)).

13: $\eta := \big(D^2f(x_0), D^2f(x_1)\big)$

$\eta_0 := \left(-\dfrac{\sqrt{A}}{4}\big(7\sqrt{A} + 3\sqrt{B}\big),\ \ \dfrac{\sqrt{B}}{4}\big(3\sqrt{A} + 7\sqrt{B}\big)\right)$

14: $\big(D^2f(x_0),\ D^2f(x_1)\big) :=$ `binary_search`$\big(g, \eta, \eta_0\big)$

Here $g(z) =$ `is_monotone`(x_0, x_1, f) where x_0, x_1, $f(x_0)$, $f(x_1)$, $Df(x_0)$, $Df(x_1)$ are fixed, and the variable $z = \big(D^2f(x_0), D^2f(x_1)\big)$. This binary search identifies feasible values of $D^2f(x_0)$ and $D^2f(x_1)$ that are nearest to the current values η. This search is valid because η_0 is guaranteed to produce a monotone function, which can be seen in Equation (23) of Ulrich and Watson (1994).

`return`

Notice that both Algorithms 1 and 2 have $\mathcal{O}(1)$ runtime assuming a fixed level of precision is chosen beforehand. A constant number of operations are needed for verifying monotonicity (Algorithm 1), while a constant set of operations and a single binary search are performed for enforcing monotonicity (Algorithm 2). The search requires a fixed number of steps to achieve any predetermined relative precision, since its accuracy is predetermined and not a function of the problem at hand. Also note that an efficient implementation of Algorithm 2 only needs to recalculate Steps 11 and 12 of Algorithm 1 during the binary search. Next, an algorithm for constructing a monotone quintic spline interpolant is presented.

Algorithm 3: `monotone_spline`$\big((k_1,\ \ldots,\ k_n),\ f,\ s\big)$
where $(k_1, \ldots, k_n)$ is an increasing sequence of real numbers, f is an order six piecewise polynomial with breakpoints $k_1, \ldots, k_n$ defined by the data $\{f(k_i)\}_{i=1}^n$, $\{Df(k_i)\}_{i=1}^n$, $\{D^2f(k_i)\}_{i=1}^n$, and $s > 1$ is an integer shrink factor.

1: `create queue`
`create changed`$(1,\ \ldots,\ n) :=$ `FALSE`
`create step_size`$(1,\ \ldots,\ n) := \big(Df(k_1)/s,\ \ldots,\ Df(k_n)/s\big)$

The three variables defined in Step 1 are used to ensure the eventual achievement of monotonicity. The `queue` *is a standard first in first out (FIFO) queue with* `enqueue` *and* `dequeue` *operations. The array* `changed` *contains Booleans describing whether or not a breakpoint belongs to an interval that has been modified to enforce monotonicity. The* `step_size` *array contains the real-valued derivative decrement step sizes to use in the search for a valid monotone spline.*

2: `for` $i := 1,\ \ldots,\ n-1$
`enqueue` $\big((k_i, k_{i+1})\big)$
`end for`

Initially, all intervals must be checked for monotonicity. The following loop will continue until all intervals are verified as monotone without need for modification.

3: `while` $(\texttt{size}(\texttt{queue}) > 0)$
4: $(k_j,\ k_{j+1}) :=$ `dequeue`
5: `if` $\big($`not is_monotone`$(k_j,\ k_{j+1},\ f)\big)$
6: `if` $\big($`changed`$(j)\big)$ $Df(k_j) := Df(k_j) - \texttt{step\_size}(j)$
`if` $\big($`changed`$(j+1)\big)$ $Df(k_{j+1}) := Df(k_{j+1}) - \texttt{step\_size}(j+1)$

If a breakpoint belongs to an interval that has been previously modified, then the enforced monotonicity conditions on the second derivatives of adjacent intervals must contradict one another. In turn, the involved first derivatives are decreased towards zero by a predetermined step size.

7: `make_monotone`$(k_j,\ k_{j+1},\ f)$
8: `changed`$(j) :=$ `TRUE`
`changed`$(j+1) :=$ `TRUE`
9: `if` $\big(j > 1$ `and` $(k_{j-1}, k_j) \notin$ `queue`$\big)$ `enqueue` $\big((k_{j-1}, k_j)\big)$
`enqueue` $\big((k_j, k_{j+1})\big)$
`if` $\big(j+1 < n$ `and` $(k_{j+1}, k_{j+2}) \notin$ `queue`$\big)$ `enqueue` $\big((k_{j+1}, k_{j+2})\big)$

Step 8 records the endpoints of the current interval as having been changed, while 9 adds adjacent intervals to `queue` *that may have inadvertently been made nonmonotone by the changes to the present interval.*

`end if`

```
    end while
```

It is mentioned in Ulrich and Watson (1994) that for sufficiently small $Df(k_i)$ and $Df(k_{i+1})$ the admissible solution interval of second derivative values becomes arbitrarily large. It can also be seen that decreasing the assigned derivative values towards zero will eventually cause Steps 5 through 7 of Algorithm 1 to all fail, resulting in Step 8 returning `TRUE`.

If successive monotonicity fixes are required for all intervals, the worst case runtime of Algorithm 3 is $\mathcal{O}(sn)$ for n breakpoints and shrink factor s. In practice this worst case has been observed to be unlikely.

3 EXPERIMENTAL RESULTS

Given the increased order of approximation, it is naturally expected that some accuracy benefit could be gained by using a quintic spline over a cubic spline. Experiments 1 and 2 below test some accuracy differences between monotone cubic and quintic splines. Experiment 3 analyzes the number of operations performed by Algorithm 3 with increasingly large samples of monotone data.

3.1 Approximating a Trigonometric Function

For this experiment, the function $\sin(x)+x$ is considered over the interval $[0,(5/2)\pi]$ as seen in Figure 3. Given an increasing number of points, the maximum error of cubic and quintic approximations is shown in Figure 4. The quintic approximation consistently has a maximum error that is roughly half that of the cubic approximation, given the same number of points. There is an increase in the error gap between the two approximations as more points are added. The quintic spline requires only 700 points to approximate the function to the same accuracy achieved by the cubic with 1000 points.

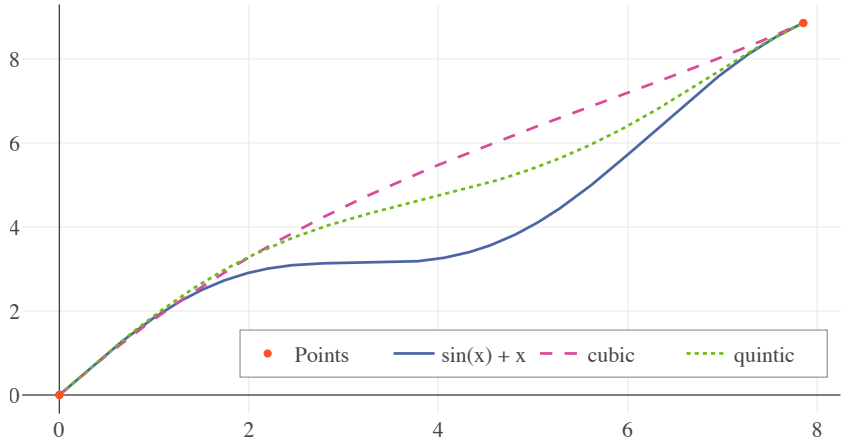

Figure 3: Depicted above are the monotone cubic and quintic spline interpolants of the function $\sin(x)+x$ over the points $[0,(5/2)\pi]$. Notice that the maximum error of the cubic interpolant is larger, because it only captures first derivative information at the points. The quintic interpolant captures both first and second derivative information at the points.

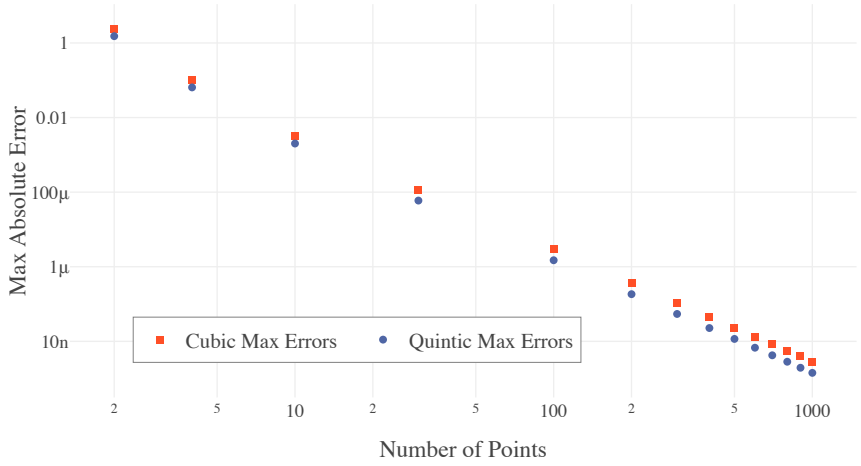

Figure 4: The error in the quintic and cubic approximations of $\sin(x) + x$ with increasing number of points. Both axes are log scaled, where μ means $\times 10^{-6}$ and n means $\times 10^{-9}$ on the vertical axis. The quintic approximation generally provides an approximation with about half of the maximum error of the cubic approximation, while the relative gap between grows with increasing number of points. In order to achieve the same absolute error the quintic spline requires ten fewer points at max error 10^{-4}, thirty fewer points at max error 10^{-6}, and 300 fewer points at max error 10^{-8}.

3.2 Random Number Generation

Consider the cumulative distribution function (CDF) defined by a mixture of three Gaussian distributions and visualized in Figure 5. The distributions have weights $(.3, .6, .1)$, means $(.2, .45, .85)$ and standard deviations $(.05, .08, .03)$. 200 random samples are generated from this distribution with both cubic and quintic approximations from four points, and over 100 trials the resulting spread of empirical distribution functions (EDFs) are visualized in Figure 6.

While an increased number of points could be used to decrease the error in either approximation, the purpose of this experiment is to demonstrate the effect approximation error has on random number generation. The errors in the CDFs are magnified by the variance involved in randomly sampling from a distribution. As a result, while the cubic and quintic spline approximations have similar maximum error, the cubic distribution is significantly distorted around the first and second modes.

3.3 Random Monotone Data

This experiment studies the number of times the algorithms `is_monotone` and `make_monotone` are executed for increasingly large sequences of random monotone data. The minimum, median, and maximum number of times that Algorithms 1 and 2 are executed is recorded, as well as the number of steps taken in all calls to `binary_search`. The number of calls and steps grows linearly with n as expected, requiring roughly one call to `make_monotone` and 20 binary search steps to create a monotone piece. Some (less common) problems require an average of three calls to `make_monotone` per interval.

4 DISCUSSION

The monotone quintic spline interpolant provides a distinct increase in accuracy over the monotone cubic variant. The computational complexity of this algorithm is identical to the existing cubic method and the

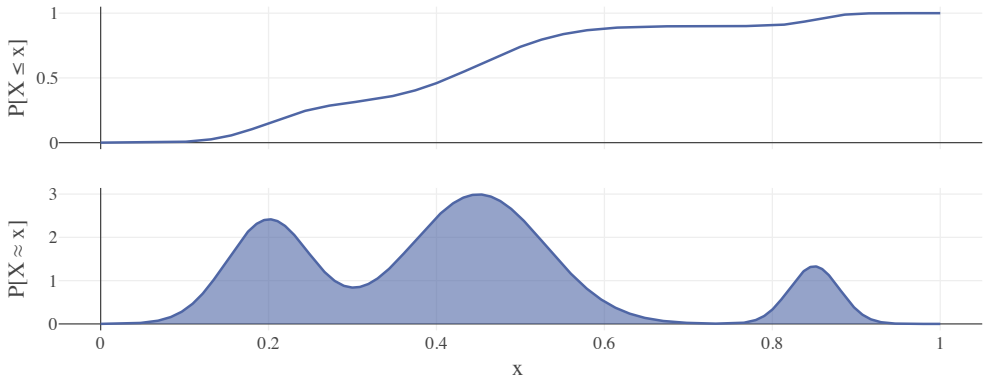

Figure 5: The cumulative distribution function (top) and probability density function (bottom) for the Gaussian mixture distribution in Experiment 2. The distributions have weights $(.3, .6, .1)$, means $(.2, .45, .85)$ and standard deviations $(.05, .08, .03)$. Random samples are generated from this mixture distribution by evaluating the inverse of the CDF at random numbers uniformly distributed in the range $[0, 1]$.

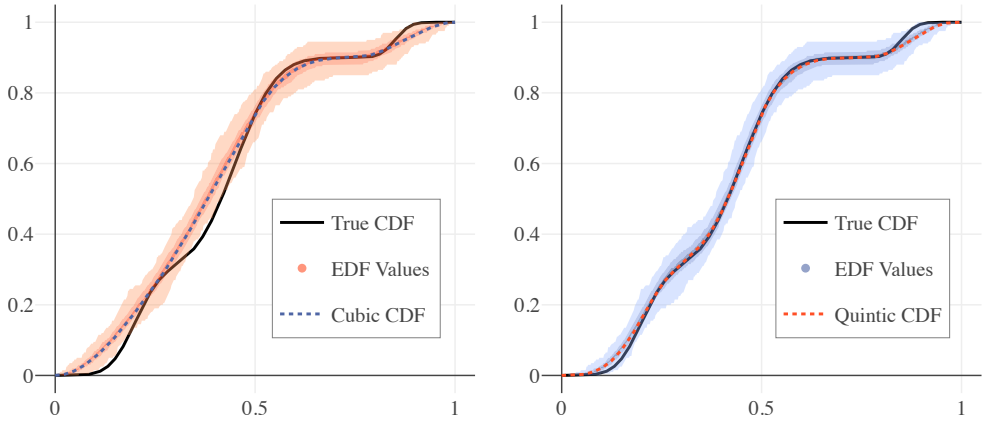

Figure 6: Both cubic and quintic splines are used to approximate the CDF of a Gaussian mixture distribution with three components. Four CDF points are used to build the approximation, 200 random samples are generated with the approximate CDFs over 100 trials. The resulting cloud of empirical distribution function (EDF) values is seen above. The maximum approximation error in the CDFs are similar, .08 for the cubic and .05 for the quintic. However, the maximum EDF differences observed are doubled by the variance of random samples, .15 for the cubic and .1 for the quintic.

n	Checks			Fixes			Search Steps		
	min	median	max	min	median	max	min	median	max
5	4	6	274	0	1	154	3	31	2006
25	28	47	378	2	14	225	80	348	3742
50	61	124	484	6	40	238	217	851	4445
75	102	216	842	14	74	411	463	1491	6333
100	142	306	776	21	107	380	629	2140	7414

Table 1: The minimum, median, and maximum number of checks, fixes, and binary search steps required in the execution of `make_spline` for increasing size sequences, n, over 100 randomly generated sets of monotone data. Notice the maximum for each counter is often significantly greater than the minimum and median, because the distribution of each counter is skewed right.

cost of evaluating the spline after construction is unchanged. While the number of computations required per interval has increased, the number of points required to achieve the same level of accuracy has decreased. In the present state, the algorithm for enforcing monotonicity on a spline is not as trivially parallel as the cubic algorithm, however it can still be parallelized. The checks for monotonicity across all intervals are independent and the monotonicity adjustments could be computed independently and intelligently merged with minor changes to the serial algorithm.

Acknowledging that the algorithm for constructing a monotone quintic spline interpolant is slightly more expensive than the cubic case, gains in accuracy or decreased necessary number of points are often worth the computational effort. In practice the costs of spline interpolation are often dominated by evaluation rather than construction, and the increased accuracy is afforded for a negligible increase in computational cost.

5 CONCLUSION AND FUTURE WORK

This paper proposes and tests an algorithm for constructing monotone quintic spline interpolants. Experiments demonstrate an improvement in approximation accuracy over monotone cubic spline interpolants, as expected based on theory. There are still open avenues of research going forward, such as an alternative sufficient condition for enforcing monotonicity or increased order monotone approximations. If the monotonicity conditions can be generalized to any order or made linear, the search for a monotone interpolating spline could potentially be formulated as a convex optimization problem. Finally, this work could be used to improve cumulative distribution function (CDF) estimates as well as predictive models that use CDFs.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation Grants CNS-1565314 and CNS-1838271.

REFERENCES

Berglund, T., A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist. 2009. “Planning smooth and obstacle-avoiding B-spline paths for autonomous mining vehicles”. *IEEE Transactions on Automation Science and Engineering* vol. 7 (1), pp. 167–172.

Brennan, A. 2019, 11. “Measure, Modulation and Metadesign: NC Fabrication in Industrial Design and Architecture”. *Journal of Design History*.

Carlson, R., and F. Fritsch. 1985. “Monotone Piecewise Bicubic Interpolation”. *SIAM Journal on Numerical Analysis* vol. 22 (2), pp. 386–400.

Fritsch, F., and R. Carlson. 1980. "Monotone Piecewise Cubic Interpolation". *SIAM Journal on Numerical Analysis* vol. 17 (2), pp. 238–246.

Gregory, J. A. 1985. "Shape preserving spline interpolation". *Brunel University*.

He, X., and P. Shi. 1998. "Monotone B-spline smoothing". *Journal of the American Statistical Association* vol. 93 (442), pp. 643–650.

Herman, Daniel Lawrence and Oftedal, Mark J 2015, December. "Techniques and workflows for computer graphics animation system". US Patent 9,216,351.

Hess, W., and J. W. Schmidt. 1994. "Positive quartic, monotone quintic C2-spline interpolation in one and two dimensions". *Journal of Computational and Applied Mathematics* vol. 55 (1), pp. 51–67.

Huynh, H. T. 1993. "Accurate Monotone Cubic Interpolation". *SIAM Journal on Numerical Analysis* vol. 30 (1), pp. 57–100.

Knott, G. D. 2012. *Interpolating cubic splines*, Volume 18. Springer Science & Business Media.

Leitenstorfer, F., and G. Tutz. 2006. "Generalized monotonic regression based on B-splines with an application to air pollution data". *Biostatistics* vol. 8 (3), pp. 654–673.

Piah, A. R. M., and K. Unsworth. 2011. "Improved sufficient conditions for monotonic piecewise rational quartic interpolation". *Sains Malaysiana* vol. 40 (10), pp. 1173–1178.

Quint, A. 2003, July. "Scalable vector graphics". *IEEE MultiMedia* vol. 10 (3), pp. 99–102.

Ramsay, J. O. et al. 1988. "Monotone Regression Splines in Action". *Statistical Science* vol. 3 (4), pp. 425–441.

Schmidt, J. W., and W. Heß. 1988, Jun. "Positivity of cubic polynomials on intervals and positive spline interpolation". *BIT Numerical Mathematics* vol. 28 (2), pp. 340–352.

Ulrich, G., and L. Watson. 1994. "Positivity Conditions for Quartic Polynomials". *SIAM Journal on Scientific Computing* vol. 15 (3), pp. 528–544.

Wang, Q., and J. Tan. 2004. "Rational quartic spline involving shape parameters". *Journal of Information and Computational Science* vol. 1 (1), pp. 127–130.

Xie, Y., C. B. King, Y. Hong, and Q. Yang. 2018. "Semiparametric models for accelerated destructive degradation test data analysis". *Technometrics* vol. 60 (2), pp. 222–234.

Yao, J., and K. E. Nelson. 2018. "An Unconditionally Monotone C2 Quartic Spline Method with Nonoscillation Derivatives". *Advances in Pure Mathematics* vol. 8 (LLNL-JRNL-742107).

AUTHOR BIOGRAPHIES

THOMAS C. H. LUX is a Ph.D. candidate in computer science at Virginia Polytechnic Institute and State University (VPI&SU) studying under Dr. Layne Watson. His email address is tchlux@vt.edu.

LAYNE T. WATSON (Ph.D., Michigan, 1974) has interests in numerical analysis, mathematical programming, bioinformatics, and data science. He has been involved with the organization of HPCS since 2000.

TYLER H. CHANG is a Ph.D. candidate at VPI&SU studying computer science under Dr. Layne Watson.

LI XU is a Ph.D. candidate at VPI&SU studying statistics under Dr. Yili Hong.

YUEYAO WANG is a Ph.D. candidate at VPI&SU studying statistics under Dr. Yili Hong.

YILI HONG (Ph.D., Iowa State, 2009) is an associate professor of Statistics at VPI&SU.