Algorithm XXXX: MQSI—Monotone Quintic Spline Interpolation

THOMAS C. H. LUX and LAYNE T. WATSON
Virginia Polytechnic Institute and State University
TYLER H. CHANG
Argonne National Laboratory
WILLIAM I. THACKER
Winthrop University

MQSI is a Fortran 2003 subroutine for constructing monotone quintic spline interpolants to univariate monotone data. Using sharp theoretical monotonicity constraints, first and second derivative estimates at data provided by a quadratic facet model are refined to produce a univariate C^2 monotone interpolant. Algorithm and implementation details, complexity and sensitivity analyses, usage information, a brief performance study, and comparisons with other spline approaches are included.

Categories and Subject Descriptors: G.1.1 [**Numerical Analysis**]: Interpolation — Spline and piecewise polynomial interpolation; J.2 [**Computer Applications**]: Physical Science and Engineering — *Mathematics*; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms, Monotonicity, Spline

Additional Key Words and Phrases: quintic spline, interpolation

---

1. INTRODUCTION

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 et al. 2015; Quint 2003], aesthetic structural support in architecture [Brennan 2020], efficient aerodynamic surfaces in automotive and aerospace engineering [Brennan 2020], 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.

This work was supported by National Science Foundation Grants CNS-1565314, CNS-1838271, and DGE-154362. Authors' addresses: T. C. H. Lux, L. T. Watson, Departments of Computer Science, Mathematics, and Aerospace and Ocean Engineering, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061; e-mails: `tchlux@vt.edu, ltwatson@computer.org`; T. H. Chang, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Bldg. 240, Lemont, IL 60439; e-mail: `thchang@vt.edu`; W. I. Thacker, Winthrop University, Rock Hill, SC 29733; `thackerw@winthrop.edu`.

It is often the case that the true underlying function or phenomenon being modeled has known properties like 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 properties the present algorithm will preserve in approximations are monotonicity and C^2 continuity. In addition to previously mentioned applications, these properties are crucially important in statistics to the approximation of a cumulative distribution function and subsequently the effective generation of random numbers from a specified distribution [Ramsay 1988]. A spline 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 produce a corresponding probability density function (PDF) that is continuously differentiable, which is desirable.

The existing research in shape preserving interpolatory splines is rich and filled with many approaches for many different applications. In the context of this work, the unique quality of shape preserving spline algorithms can be observed through: (1) the type of spline (rational or Hermite), (2) the polynomial order and level of continuity, and (3) the sharpness, sufficiency, and necessity of the conditions used to establish shape preserving properties. While the choice of spline representation is less important, achieving higher levels of continuity is often more difficult than lower levels of continuity. And while all referenced algorithms establish sufficient conditions for shape preservation, it is less common and much more difficult to provide sharp *necessary* conditions for monotonicity. The following two paragraphs highlight a collection of research related to the construction of sufficient conditions for C^1 and C^2 shape preserving splines. From this foundation, the value and utility of the code presented here can be more readily appreciated.

DeVore [1977] proves that monotone spline interpolants have improved accuracy over their nonmonotone counterparts when approximating monotone functions, while Constantini [1986] proves that monotone spline interpolants to data of fixed continuity and order exist in general. This groundwork establishes the potential for monotone spline interpolation. Schumaker [1983] and McAllister and Roulier [1981] (independently) establish necessary and sufficient conditions for monotone C^1 quadratic splines through the insertion of additional knots and Fritsch [1982] establishes simplified sufficient conditions for monotone C^1 cubic splines that require no knot insertion. Gregory and Delbourgo [1983] prove necessary and sufficient conditions for a closed form solution to monotone C^1 quadratic rational spline interpolation with nonlinear boundary equations and an iterative approach. Delbourgo and Gregory [1993] extend that same work to achieve sufficient conditions for C^2 continuity with a cubic rational spline given additional tension parameters defined by users. Huynh [1993] similarly arrives at monotone C^1 cubic spline interpolants and several necessary nonlinear boundary conditions on monotonicity.

Continuing towards monotone C^2 spline interpolation, Fiorot and Tabka [1991] prove a simple method for determining the existence of a monotone C^2 cubic interpolating spline, but note that functions of this kind do not always exist for arbitrary convex monotone data sets. Pruess [1993] proposes sufficient conditions for a cubic C^2 shape preserving spline method while acknowledging that quintic splines are necessary for generally monotone C^2 spline interpolants that do not insert new knots. Similarly, Manni [1997] constructs sufficient conditions for a monotone C^2 cubic spline by adding two additional knots per interval, then Cravery and Manni [2003] extend that methodology to arrive at monotone C^3 interpolating splines by progressively increasing tension through Bezier control polygons. Constantini [1997] provides a method for monotone C^2 quintic spline interpolation based on sufficient and necessary boundary value conditions, but the necessary conditions are not sharp. Dougherty, Edelman, and Hyman [1989] construct monotone C^2 Hermite splines and in commentary recommend against the direct optimization for smoothness parameters like global L^2, rather promoting problem specific definitions of geometric niceness. Both Wang and Tan [2004] as well as Yao and Nelson [2018] propose sufficient conditions for C^2 quartic splines, while Piah and Unsworth [2011] improve upon those sufficient conditions for C^2 quartic rational spline interpolation. Abbas, Majid, and Ali [2012] formulate a set of sufficient conditions for monotone C^2 cubic rational splines and similar work is extended to C^2 quartic rationals (and splines with arbitrary smoothness) by Zhu and Han [2015]. Ulrich and Watson [1994] arrived at sufficient and sharp necessary conditions for monotone C^2 quintic splines, as also did Heß and Schmidt [1994] (independently). In addition, a statistical method for bootstrapping the construction of an arbitrarily smooth monotone fit exists [Leitenstorfer and Tutz 2006], but this is enabled by sufficient conditions only.

The currently available peer reviewed and published software for monotone piecewise polynomial interpolation is severely reduced in comparison with the number of published approaches mentioned above. This is partially indicative of how difficult it is to properly handle the numerical conditions that arise when constructing and evaluating precise shape preserving splines, as well as how difficult it is to create code that correctly conforms with the theoretical expectations. The sufficient C^1 cubic spline method PCHIP of Fritsch and Carlson [1980] is available through the SciPy Python package ($>$ 1 billion downloads), and the C^1 quadratic method of Schumaker [1983] is available as an R package (300 thousand downloads). The piecewise quadratic C^1 method of McAllister and Roulier [1981] is available in FORTRAN (1000 downloads) and the sufficient C^2 quintic method of Costantini [1997] BVSPIS in FORTRAN as well (500 downloads). Based on publicly available download records, it is assumed that the code by Fritsch [1982] for monotone C^1 cubic spline interpolation is the predominant code for constructing monotone interpolants at present.

The theory for sufficient and *sharp* necessary bounds for monotone quintic interpolation has been provided [Ulrich and Watson 1994; Heß and Schmidt 1994] and that theory was recently utilized in a proposed algorithm [Lux 2020] for monotone quintic spline construction, however no published mathematical software exists for the quintic case based on sharp monotonicity conditions. The software presented here represents the first published software package for producing C^2 shape preserving splines based on sharp monotonicity conditions. This work improves upon the algorithm presented by Lux et al. [2020] by refactoring computations for improved numerical stability, estimating minimum magnitude second derivatives at breakpoints with a quadratic facet model, and using a binary search to reduce the magnitude of the modifications made to initial derivative estimates when constructing a monotone spline interpolant.

Overview

This work provides a Fortran 2003 subroutine `MQSI` based on the sharp necessary and sufficient conditions in Ulrich and Watson [1994] for the construction of monotone quintic spline interpolants of monotone data. Precisely, the problem is, given a strictly increasing sequence $X_1 < X_2 < \cdots < X_n$ of breakpoints with corresponding monotone increasing function values $Y_1 \leq Y_2 \leq \cdots \leq Y_n$, find a C^2 monotone increasing quintic spline $Q(x)$ with the same breakpoints satisfying $Q(X_i) = Y_i$ for $1 \leq i \leq n$. (`MQSI` actually does something slightly more general, producing $Q(x)$ that is monotone increasing (decreasing) wherever the data is monotone increasing (decreasing).)

The remainder of this paper is structured as follows: Section 2 provides the algorithms for constructing a C^2 monotone quintic spline interpolant to monotone data, Section 3 outlines the method of spline representation (B-spline basis) and evaluation, Section 4 analyzes the complexity and sensitivity of the algorithms in `MQSI`, and Section 5 presents timing data and some graphs of constructed interpolants as well as a visual comparison with existing monotone spline packages.

2. MONOTONE QUINTIC INTERPOLATION

In order to construct a monotone quintic interpolating spline, two primary problems must be solved. First, reasonable derivative values at data points need to be estimated. Second, the estimated derivative values need to be modified to enforce monotonicity on all polynomial pieces.

Fritsch and Carlson [1980] originally proposed the use of central differences to estimate derivatives, however this often leads to extra and unnecessary *wiggles* in the spline when used to approximate second derivatives. Modern shape-preserving spline implementations use a weighted harmonic mean to estimate derivative values at breakpoints [Moler 2008], however this method also yields approximations whose second derivative functions often have large local L^2 norm (approximations with large *wiggle*). In an attempt to capture the local shape of the data while minimizing wiggle, this package uses a facet model from image processing [Haralick and Watson 1981] to estimate first and second derivatives at breakpoints. Rather than picking a local linear or quadratic fit with minimal residual, this work uses a quadratic facet model that selects the local quadratic interpolant with minimum magnitude second derivative.

Algorithm 1: `QUADRATIC_FACET`$\big(X(1{:}n),\, Y(1{:}n),\, i\big)$
where $X_j,\, Y_j \in \mathbb{R}$ for $j = 1, \ldots, n$, $1 \le i \le n$, and $n \ge 3$. Returns the first and second derivative at X_i of the local quadratic interpolant with minimum magnitude second derivative. Approximate equality is denoted with $\approx$ and considers two numbers to be equal once they are within the machine precision ϵ of each other.

if $\big((i \ne 1 \land Y_i \approx Y_{i-1})$ **or** $(i \ne n \land Y_i \approx Y_{i+1})\big)$ **then return** $(0,0)$
else if $i = 1$ **then**
  $f_1 :=$ interpolant to (X_1, Y_1), (X_2, Y_2), and (X_3, Y_3).
  if $\big(Df_1(X_1)(Y_2 - Y_1) < 0\big)$ **then return** $(0,0)$
  else return $\big(Df_1(X_1), D^2 f_1\big)$
  endif
else if $i = n$ **then**
  $f_1 :=$ interpolant to (X_{n-2}, Y_{n-2}), (X_{n-1}, Y_{n-1}), and (X_n, Y_n).
  if $\big(Df_1(X_n)(Y_n - Y_{n-1}) < 0\big)$ **then return** $(0,0)$
  else return $\big(Df_1(X_n), D^2 f_1\big)$
  endif
else if $\big(1 < i < n \land (Y_{i+1} - Y_i)(Y_i - Y_{i-1}) < 0\big)$ **then**
  The point $(X_i,\, Y_i)$ is an extreme point. The quadratic with minimum magnitude second derivative that has slope zero at X_i will be the facet chosen.
  $f_1 :=$ interpolant to (X_{i-1}, Y_{i-1}), (X_i, Y_i), and $Df_1(X_i) = 0$.
  $f_2 :=$ interpolant to (X_i, Y_i), (X_{i+1}, Y_{i+1}), and $Df_2(X_i) = 0$.
  if $\big(|D^2 f_1| \le |D^2 f_2|\big)$ **then return** $\big(0,\, D^2 f_1\big)$
  else return $\big(0,\, D^2 f_2\big)$
  endif
else
  The point $(X_i,\, Y_i)$ is in a monotone segment of data. In the following, it is possible that f_1 or f_3 do not exist because $i \in \{2, n-1\}$. In those cases, the minimum magnitude second derivative among existing quadratics is chosen.
  $f_1 :=$ interpolant to (X_{i-2}, Y_{i-2}), (X_{i-1}, Y_{i-1}), and (X_i, Y_i).
  $f_2 :=$ interpolant to (X_{i-1}, Y_{i-1}), (X_i, Y_i), and (X_{i+1}, Y_{i+1}).
  $f_3 :=$ interpolant to (X_i, Y_i), (X_{i+1}, Y_{i+1}), and (X_{i+2}, Y_{i+2}).
  if $\big(Df_1(X_i)(Y_i - Y_{i-1}) \ge 0 \land |D^2 f_1| = \min\big\{|D^2 f_1|, |D^2 f_2|, |D^2 f_3|\big\}\big)$ **then**
    return $\big(Df_1(X_i), D^2 f_1\big)$
  else if $\big(Df_2(X_i)(Y_i - Y_{i-1}) \ge 0 \land |D^2 f_2| = \min\big\{|D^2 f_1|, |D^2 f_2|, |D^2 f_3|\big\}\big)$
    then return $\big(Df_2(X_i), D^2 f_2\big)$
  else if $\big(Df_3(X_i)(Y_{i+1} - Y_i) \ge 0\big)$ **then**
    return $\big(Df_3(X_i), D^2 f_3\big)$
  else return $(0,0)$
  endif
endif

For constructing a quadratic interpolant in x over the interval $[L, R]$, the Chebyshev basis 1, z, $2z^2 - 1$ is used, where $z = \frac{x-(L+R)/2}{(R-L)/2}$. The estimated derivative values by the quadratic facet model are not guaranteed to produce monotone quintic polynomial segments. Ulrich and Watson [1994] established tight constraints on the monotonicity of a quintic polynomial piece, while deferring to Heß and Schmidt [1994] for a relevant simplified case. The following algorithm implements a sharp check for monotonicity by considering the nondecreasing case. The nonincreasing case is handled similarly.

Algorithm 2: `IS_MONOTONE`$\big(x_0,\ x_1,\ f(x_0),\ Df(x_0),\ D^2f(x_0),\ f(x_1),\ Df(x_1),\ D^2f(x_1)\big)$

where $x_0,\ x_1 \in \mathbb{R}$, $x_0 < x_1$, and f is an order six polynomial defined by $f(x_0)$, $Df(x_0)$, $D^2f(x_0)$, $f(x_1)$, $Df(x_1)$, $D^2f(x_1)$. Returns `TRUE` if f is monotone increasing on $[x_0, x_1]$. Approximate equality is denoted with $\approx$ and considers two numbers to be equal once they are within the machine precision ϵ of each other.

1. `if` $\big(f(x_0) \approx f(x_1)\big)$ `then`
2. `return` $\big(0 = Df(x_0) = Df(x_1) = D^2f(x_0) = D^2f(x_1)\big)$
3. `endif`
4. `if` $\big(Df(x_0) < 0$ `or` $Df(x_1) < 0\big)$ `then return FALSE endif`
5. $w := x_1 - x_0$
6. $z := f(x_1) - f(x_0)$

   The necessity of Steps 1–4 follows directly from the fact that f is C^2. The following Steps 7–13 coincide with a simplified condition for quintic monotonicity that reduces to one of cubic positivity studied by Schmidt and Heß [1988]. Given α, β, γ, and δ as defined by Schmidt and Heß, monotonicity results when $\alpha \geq 0$, $\delta \geq 0$, $\beta \geq \alpha - 2\sqrt{\alpha\delta}$, and $\gamma \geq \delta - 2\sqrt{\alpha\delta}$. Step 4 checked for $\delta < 0$, Step 8 checks $\alpha < 0$, Step 10 checks $\beta < \alpha - 2\sqrt{\alpha\delta}$, and Step 11 checks $\gamma < \delta - 2\sqrt{\alpha\delta}$. If none of the monotonicity conditions are violated, then the degree five piece is monotone and Step 12 concludes.

7. `if` $\big(Df(x_0) \approx 0$ `or` $Df(x_1) \approx 0\big)$ `then`
8. `if` $\big(D^2f(x_1)w > 4Df(x_1)\big)$ `then return FALSE endif`
9. $t := 2\sqrt{Df(x_0)(4Df(x_1) - D^2f(x_1)w)}$
10. `if` $\big(t + 3Df(x_0) + D^2f(x_0)w < 0\big)$ `then return FALSE endif`
11. `if` $\Big(60z - w\big(24Df(x_0) + 32Df(x_1) - 2t + w(3D^2f(x_0) - 5D^2f(x_1))\big) < 0\Big)$ `then return FALSE endif`
12. `return TRUE`
13. `endif`

    The following code considers the full quintic monotonicity case studied by Ulrich and Watson [1994]. Given τ_1, α, β, and γ as defined by Ulrich and Watson, a quintic piece is proven to be monotone if and only if $\tau_1 > 0$, and $\alpha, \gamma > -(\beta + 2)/2$ when $\beta \leq 6$, and $\alpha, \gamma > -2\sqrt{\beta - 2}$ when $\beta > 6$. Step 14 checks $\tau_1 \leq 0$, Steps 19 and 20 determine monotonicity based on α, β, and γ.

14. `if` $\Big(w\big(2\sqrt{Df(x_0)\,Df(x_1)} - 3(Df(x_0) + Df(x_1))\big) - 24z \leq 0\Big)$

then return FALSE endif
15. $t := \big(Df(x_0)\,Df(x_1)\big)^{3/4}$
16. $\alpha := (4Df(x_1) - D^2f(x_1)w)\sqrt{Df(x_0)}/t$
17. $\gamma := (4Df(x_0) - D^2f(x_0)w)\sqrt{Df(x_1)}/t$
18. $\beta := \dfrac{60z/w + 3\big(w(D^2f(x_1) - D^2f(x_0)) - 8(Df(x_0) + Df(x_1))\big)}{2\sqrt{Df(x_0)\,Df(x_1)}}$
19. `if` $(\beta \le 6)$ `then return` $\big(\min\{\alpha, \gamma\} > -(\beta + 2)/2\big)$
20. `else return` $\big(\min\{\alpha, \gamma\} > -2\sqrt{\beta - 2}\,\big)$
21. `endif`

It is shown by Ulrich and Watson [1994] that when $0 = DQ(X_i) = DQ(X_{i+1}) = D^2Q(X_i) = D^2Q(X_{i+1})$, the quintic polynomial over $[X_i,\, X_{i+1}]$ is guaranteed to be monotone. Using this fact, the following algorithm shrinks (in magnitude) initial derivative estimates until a monotone spline is achieved and outlines the core routine in the accompanying package.

Algorithm 3: `MQSI`$\big(X(1{:}n), Y(1{:}n)\big)$
where $(X_i, Y_i) \in \mathbb{R} \times \mathbb{R}$, $i = 1, \ldots, n$ are data points. Returns monotone quintic spline interpolant $Q(x)$ such that $Q(X_i) = Y_i$ and is monotone increasing (decreasing) on all intervals that Y_i is monotone increasing (decreasing).

Approximate first and second derivatives at X_i with `QUADRATIC_FACET`.
`for` $i := 1$ `step` 1 `until` n `do`
  $(u_i,\, v_i) :=$ `QUADRATIC_FACET`$(X,\, Y,\, i)$
`enddo`
Identify and store all intervals where Q is nonmonotone in a queue $\mathcal{Q}$.
`for` $i := 1$ `step` 1 `until` $n-1$ `do`
  `if not IS_MONOTONE`$(X_i,\, X_{i+1},\, Y_i,\, u_i,\, v_i,\, Y_{i+1},\, u_{i+1},\, v_{i+1})$ `then`
    Add interval $\big(X_i,\, X_{i+1}\big)$ to queue $\mathcal{Q}$.
  `endif`
`enddo`
`do while` (queue $\mathcal{Q}$ of intervals is nonempty)
  Shrink (in magnitude) DQ (in u) and D^2Q (in v) that border intervals where Q is nonmonotone.
  Identify and store remaining intervals where Q is nonmonotone in queue $\mathcal{Q}$.
`enddo`
Construct and return a B-spline representation of $Q(x)$.

Since `IS_MONOTONE` can handle both nondecreasing and nonincreasing simultaneously by taking into account the sign of z, Algorithm 3 produces $Q(x)$ that is monotone increasing (decreasing) over exactly the same intervals that the data $(X_i$, $Y_i)$ is monotone increasing (decreasing).

Given the minimum curvature (minimum magnitude of the second derivative) nature of the initial derivative estimates, it is desirable to make the smallest necessary changes to the initial interpolating spline Q while enforcing monotonicity. In practice a binary search for the boundary of monotonicity is used in place of solely shrinking DQ and D^2Q at breakpoints adjoining *active* intervals: intervals over which Q is nonmonotone at least once during the search. The binary search considers a Boolean function $B_i(s)$, for $0 \le s \le 1$, that is true if the order six polynomial piece of $Q(x)$ on $[X_i, X_{i+1}]$ matching derivatives $Q(X_i) = Y_i$, $DQ(X_i) = s\,u_i$, $D^2Q(X_i) = s\,v_i$ at X_i, and derivatives $Q(X_{i+1}) = Y_{i+1}$, $DQ(X_{i+1}) = s\,u_{i+1}$, $D^2Q(X_{i+1}) = s\,v_{i+1}$ at X_{i+1} is monotone, and false otherwise. The binary search is only applied at those breakpoints adjoining intervals $[X_i, X_{i+1}]$ over which Q is nonmonotone and hence $B_i(1)$ is false. It is further assumed that there exists $0 \le s^* \le 1$ such that $B_i(s)$ is true for $0 \le s \le s^*$ and false for some $1 > s > s^*$. Since the derivative conditions at interior breakpoints are shared by intervals left and right of the breakpoint, the binary search is performed at all breakpoints simultaneously. Specifically, the monotonicity of Q is checked on all active intervals in each step of the binary search to determine the next derivative modification at each breakpoint. The goal of this search is to converge on the boundary of the monotone region in the (τ_1, α, β, γ) space (described in Ulrich and Watson [1994]) for all intervals. This multiple-interval binary search allows the value zero to be obtained for all (first and second) derivative values in a fixed maximum number of computations, hence has no effect on computational complexity order. This binary search algorithm is outlined below.

Algorithm 4: `BINARY_SEARCH`$\big(X(1{:}n), Y(1{:}n), u(1{:}n), v(1{:}n)\big)$
where $(X_i, Y_i) \in \mathbb{R} \times \mathbb{R}$, $i = 1, \ldots, n$ are data points, and $Q(x)$ is a quintic spline interpolant such that $Q(X_i) = Y_i$, $DQ(X_i) = u_i$, $D^2Q(X_i) = v_i$. Modifies derivative values (u and v) of Q at data points to ensure `IS_MONOTONE` is true for all intervals defined by adjacent data points, given a desired precision $\mu \in \mathbb{R}$. $\mathrm{proj}\big(w, \mathrm{int}(a,b)\big)$ denotes the projection of w onto the closed interval with endpoints a and b.

Initialize the step size s, make a copy of data defining Q, and construct three queues necessary for the multiple-interval binary search.
$s := 1$
$(\hat{u}, \hat{v}) := (u, v)$
`searching` := `TRUE`
`checking` := empty queue for holding left indices of *intervals*
`growing` := empty queue for holding indices of *data points*
`shrinking` := empty queue for holding indices of *data points*
`for` $i := 1$ `step` 1 `until` $n-1$ `do`
  `if not IS_MONOTONE`$\big(X_i,\, X_{i+1},\, Y_i,\, u_i,\, v_i,\, Y_{i+1},\, u_{i+1},\, v_{i+1}\big)$ `then`
    Add data indices i and $i+1$ to queue `shrinking`.
  `endif`
`enddo`
`do while` $\big($`searching or` (`shrinking` is nonempty)$\big)$
  Compute the *step size* s for this iteration of the search.

```
    if searching then s := max{μ, s/2} else s := 3s/2 endif
    if (s = μ) then searching := FALSE; clear queue growing endif
    Increase in magnitude u_i and v_i for all data indices i in growing such that
    the points X_i are strictly adjoining intervals over which Q is monotone.
    for (i ∈ growing) and (i ∉ shrinking) do
      u_i := proj(u_i + s û_i, int(0, û_i))
      v_i := proj(v_i + s v̂_i, int(0, v̂_i))
      Add data indices i − 1 (if not 0) and i (if not n) to queue checking.
    enddo
    Decrease in magnitude u_i and v_i for all data indices i in shrinking and
    ensure those data point indices are placed into growing when searching.
    for i ∈ shrinking do
      If searching, then add index i to queue growing if not already present.
      u_i := proj(u_i − s û_i, int(0, û_i))
      v_i := proj(v_i − s v̂_i, int(0, v̂_i))
      Add data indices i − 1 (if not 0) and i (if not n) to queue checking.
    enddo
    Empty queue shrinking, then check all intervals left-indexed in queue
    checking for monotonicity with IS_MONOTONE, placing data endpoint in-
    dices of intervals over which Q is nonmonotone into queue shrinking.
    Clear queue shrinking.
    for i ∈ checking do
      if not IS_MONOTONE(X_i, X_{i+1}, Y_i, u_i, v_i, Y_{i+1}, u_{i+1}, v_{i+1}) then
        Add data indices i and i + 1 to shrinking.
      endif
    enddo
    Clear queue checking.
  enddo
```

In the subroutine MQSI, $\mu = 2^{-26}$, which results in 26 guaranteed search steps for all intervals that are initially nonmonotone. An additional 43 steps could be required to reduce a derivative magnitude to zero with step size growth rate of 3/2. This can only happen when Q becomes nonmonotone on an interval for the first time while the step size equals μ, but for which the only viable solution is a derivative value of zero. The maximum number of steps is due to the fact that $\sum_{i=0}^{42} \mu(3/2)^i > 1$. In total BINARY_SEARCH search could require 69 steps.

3. SPLINE REPRESENTATION

The monotone quintic spline interpolant $Q(x)$ is represented in terms of a B-spline basis. The routine FIT_SPLINE in this package computes the B-spline coefficients α_i of $Q(x) = \sum_{i=1}^{3n} \alpha_i B_{i,6,t}(x)$ to match the piecewise quintic polynomial values and (first two) derivatives at the breakpoints X_i, where the spline order is six and the knot sequence t has the breakpoint multiplicities (6, 3, …, 3, 6). The routine

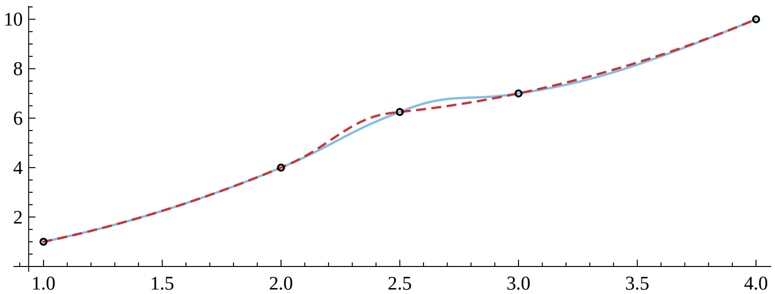

Fig. 1. A demonstration of the quadratic facet model's sensitivity to small data perturbations. This example is composed of two quadratic functions $f_1(x)=x^2$ over points $\{1, 2, 5/2\}$, and $f_2(x)=(x-2)^2+6$ over points $\{5/2, 3, 4\}$. Notably, $f_1(5/2)=f_2(5/2)$ and f_1, f_2 have the same curvature (equal second derivatives). Given the exact five data points seen above, the quadratic facet model produces the slope seen in the solid blue line at $x=5/2$. However, by subtracting the value of $f_3 = \epsilon(x-2)^2$ from points at $x=3$, 4, where ϵ is the machine precision (2^{-52} for an IEEE 64-bit real), the quadratic facet model produces the slope seen in the dashed red line at $x=5/2$. This is the nature of a facet model and a side effect of associating data with local facets.

EVAL_SPLINE evaluates a spline represented in terms of a B-spline basis. A Fortran 2003 implementation EVAL_BSPLINE of the B-spline recurrence relation evaluation code by C. de Boor [1978] for the value, derivatives, and integral of a B-spline is also provided.

4. COMPLEXITY AND SENSITIVITY

Algorithms 1 and 4 have $\mathcal{O}(n)$ runtime for n data points. Algorithm 2 has a fixed cost $\mathcal{O}(1)$. Given a fixed schedule for shrinking derivative values, Algorithm 3 has a $\mathcal{O}(n)$ runtime for n data points. In execution, the majority of the time, still $\mathcal{O}(n)$, is spent solving the banded linear system of equations for the B-spline coefficients. Thus for n data points, the overall execution time is $\mathcal{O}(n)$.

The quadratic facet model produces a unique sensitivity to input perturbation, as small changes in input may cause different quadratic facets to be associated with a breakpoint, and thus different initial derivative estimates can be produced. This phenomenon is depicted in Figure 1. Despite this sensitivity, the quadratic facet model is still preferred because it exactly captures local linear and quadratic behavior while empirically producing final approximations with less wiggle (local L^2 norm of the second derivative) than other methods. A weighted harmonic mean estimate of first derivatives may be more accurate when the underlying function changes at a rate greater than a quadratic, but that method increases the second derivative sensitivity to small perturbations in data and empirically results in quintic splines with greater wiggle.

The binary search for a point on the monotone boundary in $(\tau_1, \alpha, \beta, \gamma)$ space is performed because it results in monotone quintic spline interpolants with derivative values that are absolutely nearer to initial estimates than a search that strictly shrinks derivative values. Given that the initial derivative estimates have desirable properties (capture low-order phenomena and are low wiggle), this search results in an approximation that is both monotone and has derivative values similar to the initial estimates.

5. PERFORMANCE AND APPLICATIONS

This section contains graphs of sample MQSI results given various data configurations. Execution times for Algorithms 1 and 4 are given; the total execution time of MQSI is utterly dominated by the time for a banded linear system solve computing the $3n$ B-spline coefficients, which is $\mathcal{O}(n)$. The files sample_main.f90 and sample_main.dat accompanying the subroutine MQSI illustrate Fortran 2003 subroutine usage with optional arguments and data points from a file. Compilation instructions and the full package contents are specified in the README file.

Throughout, all visuals have points that are stylized by local monotonicity conditions. Blue circles denote extreme points, purple squares are in *flat* regions with no change in function value, red down triangles are monotone decreasing, and green up triangles are monotone increasing.

Figure 2 offers examples of the interpolating splines produced by the routine MQSI on various hand-crafted sets of data. These same data sets are used for testing local installations in the provided program test_all.f90. Notice that the quadratic facet model perfectly captures the local linear segments of data in the piecewise polynomial test for Figure 2. Figure 3 depicts an approximation of a cumulative distribution function made by MQSI on a computer systems application by Cameron et al. [2019] that studies the distribution of throughput (in bytes per second) when reading files from a storage medium. Figure 4 provides a particularly difficult monotone interpolation challenge using randomly generated monotone data.

On a computer running MacOS 10.15.5 with a 2 GHz Intel Core i5 CPU, the quadratic facet (Algorithm 1) takes roughly one microsecond (10^{-6} seconds) per breakpoint, while the binary search (Algorithm 4) takes roughly four microseconds per breakpoint; these times were generated from 100 repeated trials averaged over 14 different testing functions. The vast majority of execution time is spent solving the banded linear system of equations in the routine FIT_SPLINE for the B-spline coefficients. For large problems ($n > 100$) it would be faster to construct splines over intervals independently (each interval requiring a 6×6 linear system to be solved for a local B-spline representation, or the construction of the Newton form of the interpolating polynomial from the derivative information at the endpoints), however the single linear system is chosen here for the decreased redundancy in the spline description. An optional argument to MQSI returns the derivative information at the breakpoints (interval endpoints) for the Newton form of the interpolating polynomial (piece of $Q(x)$) over each interval.

The binary search procedure provably converges on the boundary of the region of monotonicity precisely defined by Ulrich and Watson [1994] through the application of a boolean function that is guaranteed to be true at one end of an interval. Since a local quadratic interpolant is applied for initial function derivative estimates, the approximation order of the resulting fit is $O(h^3)$ (as for any second order approximation), but it should be noted that asymptotic approximation order is rarely of import when considering the scattered sparse data that C^2 approximations like MQSI provide. Were exact evaluations provided, higher order methods always win

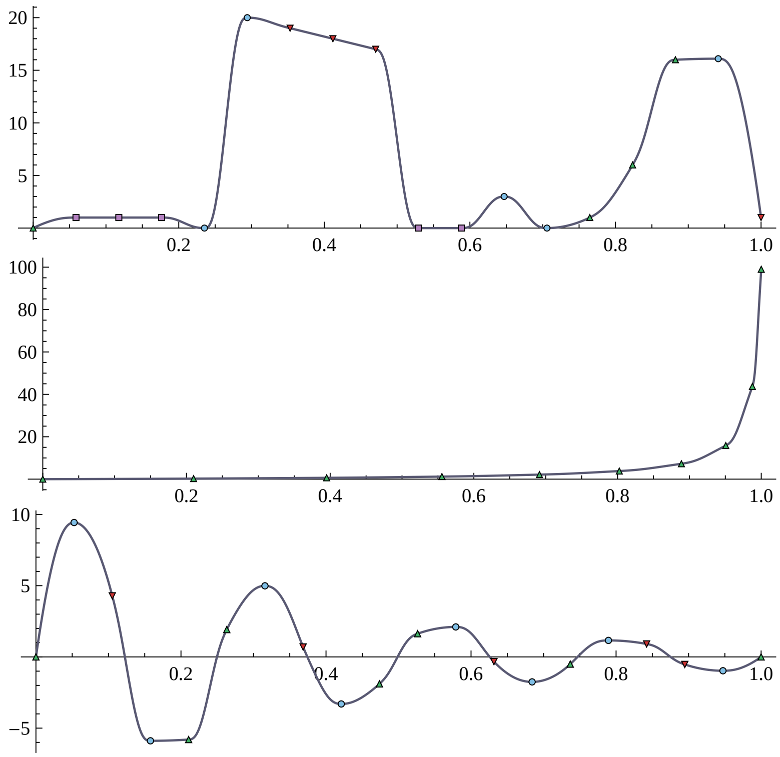

Fig. 2. `MQSI` results for three of the functions in the included test suite. The *piecewise polynomial* function (top) shows the interpolant capturing local linear segments, local flats, and alternating extreme points. The *large tangent* (middle) problem demonstrates outcomes on rapidly changing segments of data. The *signal decay* (bottom) alternates between extreme values of steadily decreasing magnitude.

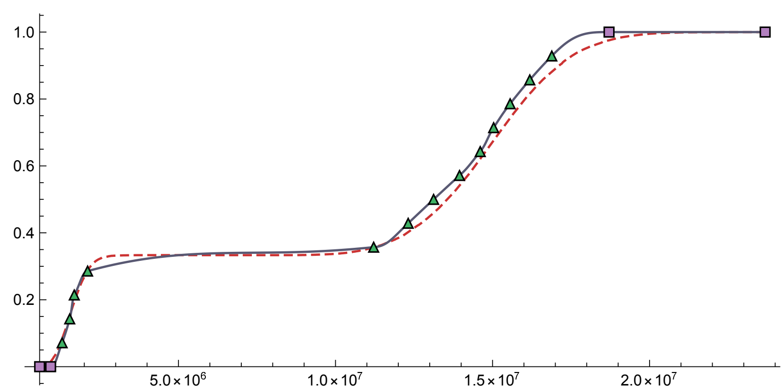

Fig. 3. `MQSI` results when approximating the cumulative distribution function of system throughput (bytes per second) data for a computer with a 3.2 GHz CPU performing file read operations from Cameron et al. [2019]. The empirical distribution of 30 thousand throughput values is shown in the red dashed line, while the solid line with stylized markers denotes the approximation made with `MQSI` given equally spaced empirical distribution points from a sample of size 100.

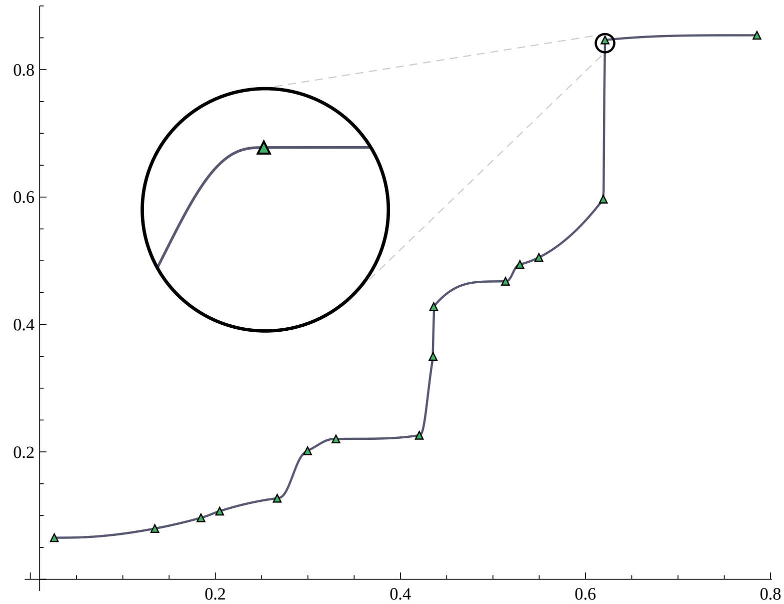

Fig. 4. The *random monotone* test poses a particularly challenging problem with large variations in slope. Notice that despite drastic shifts in slope, the resulting monotone quintic spline interpolant provides smooth and reasonable estimates to function values between data.

(all other things being equal) as the data density increases. For sparse data, what constitutes a better fit is either subjective or dependent on the problem.

Lastly, comparisons between `MQSI` and four published spline software packages are provided that allow subjective inspection on the same four test problems as presented above. In each of Figures 5 through 8 the intricacies of the approximation created by `MQSI` can be contrasted with two monotone C^1 quadratic splines (TOMS 574, Schumaker), the popular C^1 cubic method `PCHIP`, and the C^2 quintic method `BVSPIS` of TOMS 770. Figures 5 and 8 demonstrate glaring numerical errors in the Schumaker and `BVSPIS` implementations, and overall it can be observed that the minimal and tight conditions on monotonicity provided by TOMS 574 tend to be more visually appealing than the weaker sufficient conditions utilized by `PCHIP`. In general the major determining factor for the quality of `MQSI` is the local quadratic facet model. For applications where the underlying function is presumed to be a piecewise polynomial (of relatively low order), the choice of local quadratic interpolants is reasonable. However the consequence of this decision is that functions with superquadratic rates of change will tend to have consistently underestimated first and second derivatives. This limitation is accepted in favor of perfectly capturing lower order phenomenon. Were future research to apply alternative initializations for C^2 quintic splines, those could comfortably be made monotone by the procedures of `MQSI` and most importantly by the application of these sharp monotonicity conditions.

If an application demands C^2 continuity and monotonicity, the target of `MQSI`, then this package uniquely provides such interpolants based on sharp monotonicity constraints.

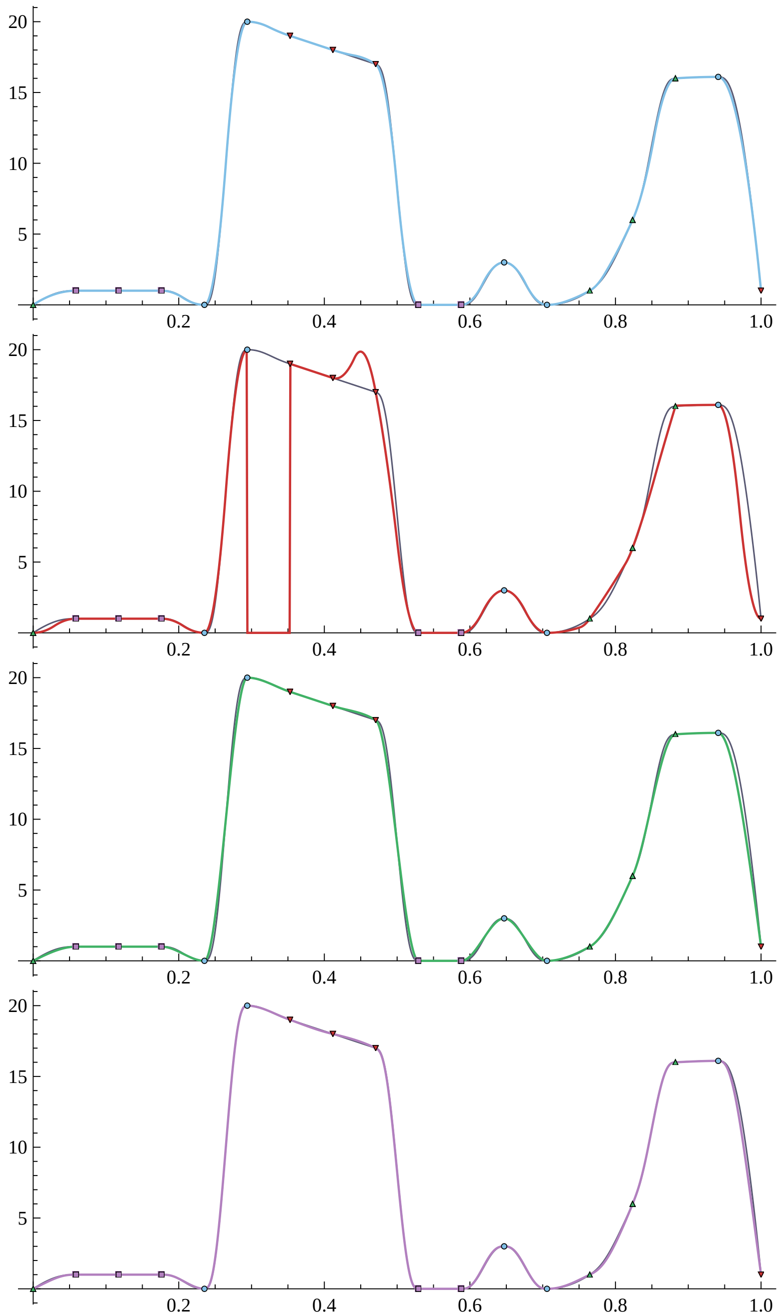

Fig. 5. MQSI compared with each of TOMS 574 (first, blue), Schumaker (second, red), PCHIP (third, green), and BVSPIS (fourth, purple) respectively on the *piecewise polynomial* test function. MQSI is styled as a gray thin line in the background for comparison. The Schumaker code (second) fails to produce a monotone approximation for this problem while producing no errors or warnings. The primary comparative observation of MQSI otherwise is its exact reproduction of the linear segment between 0.3 and 0.5.

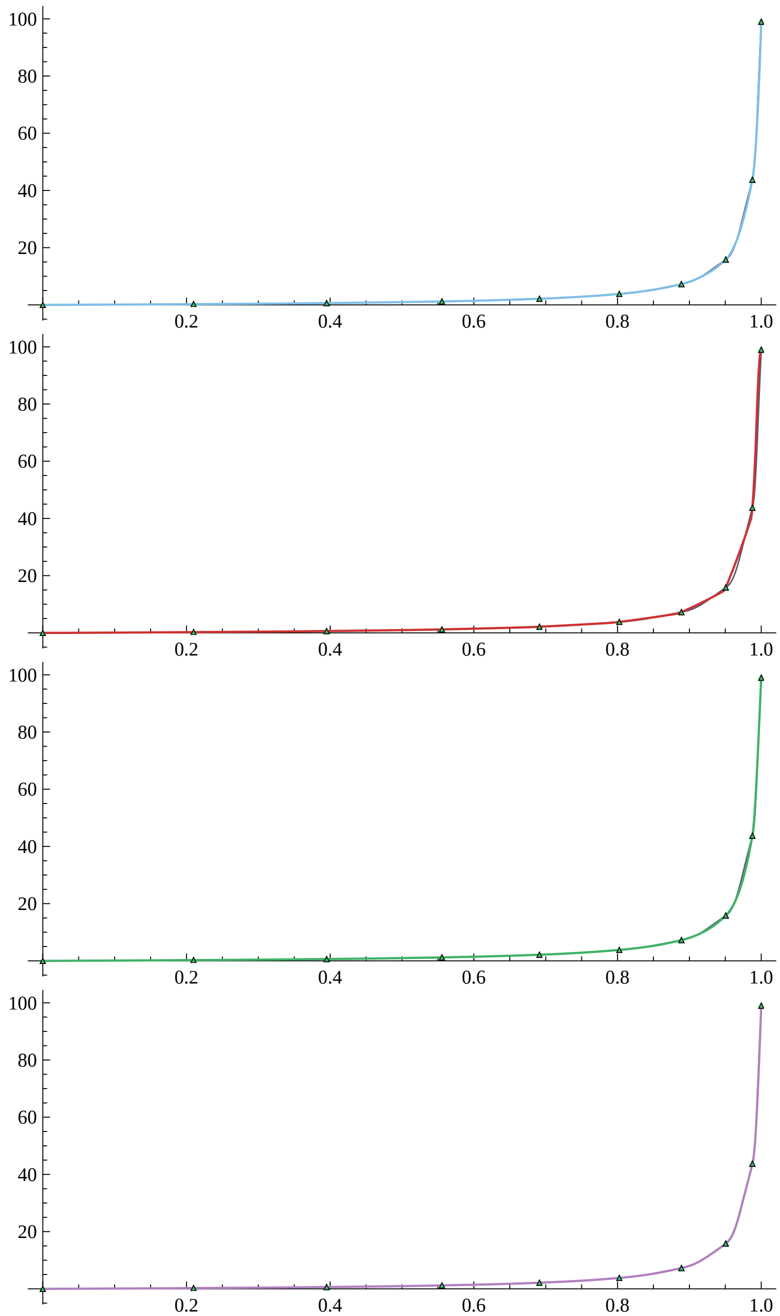

Fig. 6. MQSI compared with each of TOMS 574 (first, blue), Schumaker (second, red), PCHIP (third, green), and BVSPIS (fourth, purple) respectively on the *large tangent* test function. MQSI is styled as a gray thin line in the background for comparison. This problems highlights the core weakness of the quadratic facet model approach to derivative estimation in MQSI, which consistently underestimates the actual curvature of this function. This weakness is accepted for its greater accuracy when estimating local lower order components of functions.

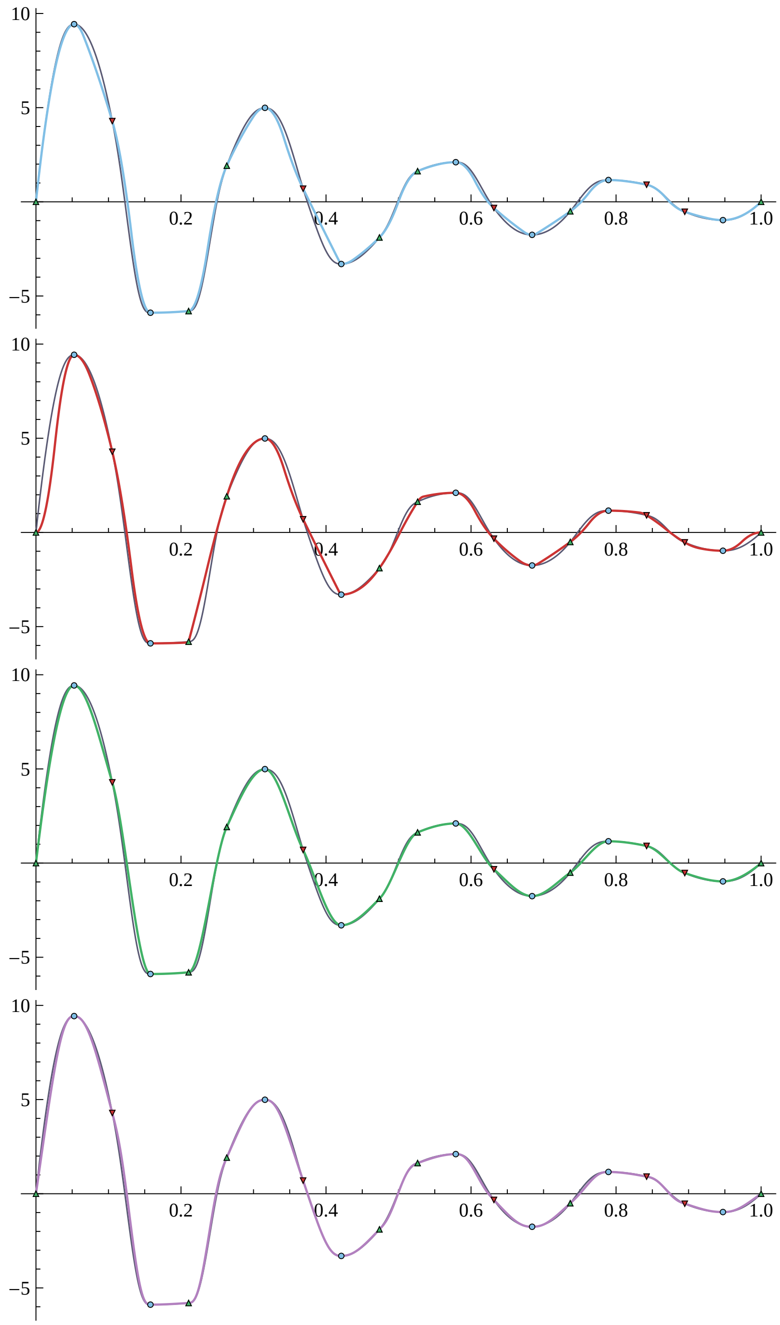

Fig. 7. MQSI compared with each of TOMS 574 (first, blue), Schumaker (second, red), PCHIP (third, green), and BVSPIS (fourth, purple) respectively on the *signal decay* test function. MQSI is styled as a gray thin line in the background for comparison. The most notable differences between MQSI and other approaches can be observed near local extrema, where MQSI produces smaller magnitude second derivatives.

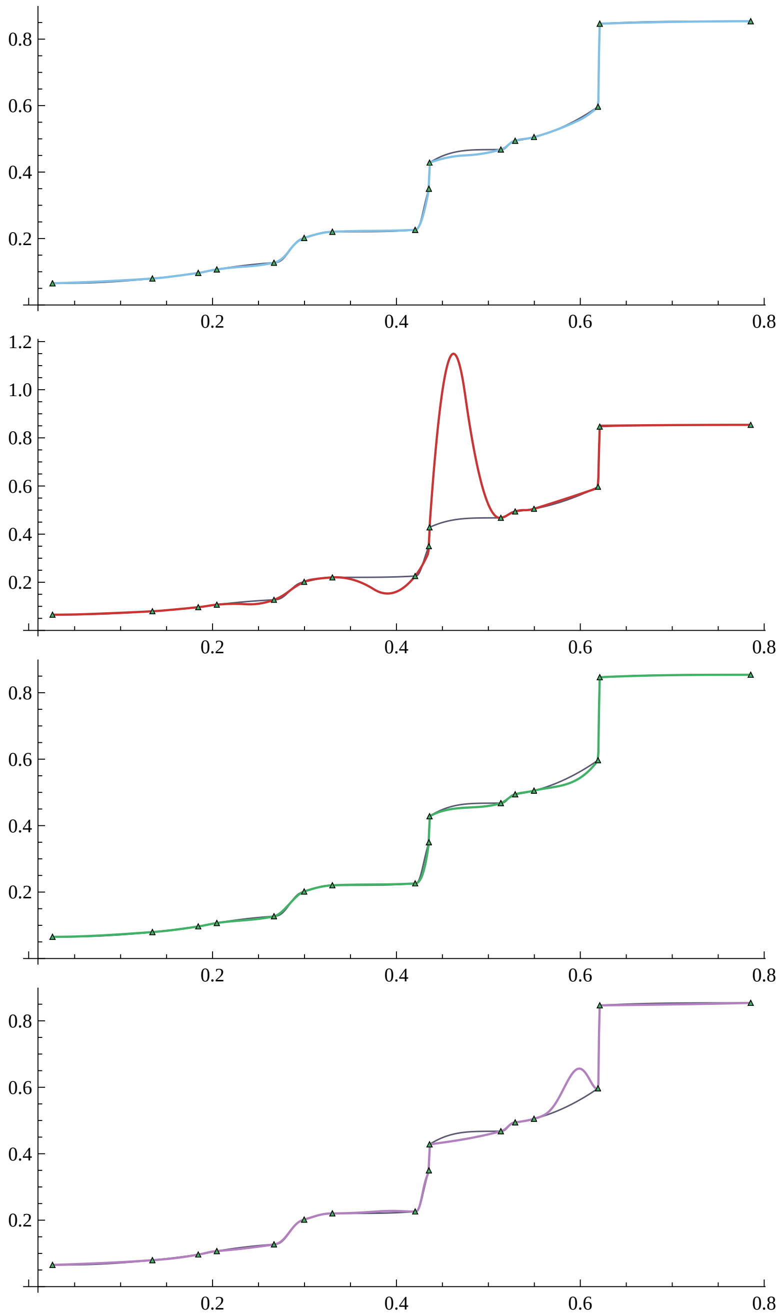

Fig. 8. `MQSI` compared with each of TOMS 574 (first, blue), Schumaker (second, red), `PCHIP` (third, green), and `BVSPIS` (fourth, purple) respectively on the *random monotone* test function. `MQSI` is styled as a gray thin line in the background for comparison. Notice that the numerical conditions for this approximation problem are challenging enough that both the Schumaker and `BVSPIS` codes incorrectly produce nonmonotone segments (neither of which produced error codes or any form of indication that a failure occurred). Notably only `MQSI` and TOMS 574 satisfy tight monotonicity constraints while also correctly preserving monotonicity in their approximations. The main segment of divergence between the methods happens on the $(0.45, 0.5)$ interval, where `MQSI` favors maximizing the slope on the left side of the interval because the quadratic interpolant of the right sided points produces an (invalid) negative slope estimate on the left side of the interval.

BIBLIOGRAPHY

Abbas M., Majid, A. A. and Ali, J. M. 2012. Monotonicity-preserving C^2 rational cubic spline for monotone data. *Applied Mathematics and Computation*, 219(6):2885–2895.

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

Brennan, A. 2020. Measure, modulation and metadesign: NC fabrication in industrial design and architecture. *Journal of Design History*, 33(1):66–82.

de Boor, C. 1978. *A Practical Guide to Splines.* Springer, Verlag, New York.

Cameron, K. W., Anwar, A., Cheng, Y., Xu, L., Li, B., Ananth, U., Bernard, J., Jearls, C., Lux, T. C. H., Hong, Y., Watson, L. T., and Butt, A. R. 2019. MOANA: modeling and analyzing I/O variability in parallel system experimental design. *IEEE Transactions on Parallel and Distributed Systems*, 30(8):1843–1856.

Costantini, P. 1986. On monotone and convex spline interpolation. *Mathematics of Computation*, 46(173):203–214.

Costantini, P. 1997. Boundary-valued shape-preserving interpolating splines. *ACM Transactions on Mathematical Software (TOMS)*, 23(2):229–251.

Costantini, P. 1997. Algorithm 770: BVSPISa package for computing boundary-valued shape-preserving interpolating splines. *ACM Transactions on Mathematical Software (TOMS)*, 23(2):252-254.

Cravero, I., C. Manni 2003. Shape-preserving interpolants with high smoothness. *Journal of computational and applied mathematics*, 157(2):383–405.

Delbourgo, R., Gregory, J. A. 1983. C^2 rational quadratic spline interpolation to monotonic data. *IMA Journal of Numerical Analysis*, 3(2):141–152.

Delbourgo, R. 1993. Accurate C^2 rational interpolations in tension. *SIAM Journal on Numerical Analysis*, 30(2):595–607.

DeVore, R. A. 1977. Monotone approximation by splines. *SIAM Journal on Mathematical Analysis*, 8(5):891–905.

DeVore, R. A. 1977. Monotone approximation by polynomials. *SIAM Journal on Mathematical Analysis*, 8(5):906–921.

Dougherty, R. L., Edelman, A., Hyman, J. M. 1989. Nonnegativity-, monotonicity- convexity-preserving cubic and quintic Hermite interpolation. *Mathematics of Computation*, 52(186):471–494.

Fiorot, J. C., Tabka, J. 1991. Shape-preserving C^2 cubic polynomial interpolating splines. *Mathematics of Computation*, 57(195):291–298.

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

Fritsch, F. N. 1982. Piecewise cubic Hermite interpolation package, final specifications. Computer Documentation UCID-30194, Lawrence Livermore National Laboratory.

Gregory , J. A.,Delbourgo, R. 1982. Piecewise rational quadratic interpolation to monotonic data. *IMA Journal of Numerical Analysis*, 2(2):123–130.

Gregory, J. A. 1985. Shape preserving spline interpolation. 19850020252, Brunel University.

Han, X. 2018. Shape-preserving piecewise rational interpolation with higher order continuity. *Applied Mathematics and Computation*, 337:1–13.

HERMAN, D. L., AND OFTEDAL, M. J. 2015. Techniques and workflows for computer graphics animation systems. US Patent 9,216,351.

HESS, W. AND SCHMIDT, J. W. 1994. Positive quartic, monotone quintic C^2-spline interpolation in one and two dimensions. *Journal of Computational and Applied Mathematics*, 55(1):51–67.

HUYNH, H. T. 1993. Accurate monotone cubic interpolation. *SIAM Journal of Numerical Analysis*, 30(1):57–100.

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

LEITENSTORFER, F. AND TUTZ, G. 2006. Generalized monotonic regression based on B-splines with an application to air pollution data. *Biostatistics*, 8(3):654–673.

LUX, T. C. H. 2020. Interpolants and error bounds for modeling and predicting variability in computer systems. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

MANNI, C., SABLONNIÈRE, P. 1997. Monotone interpolation of order 3 by C^2 cubic splines. *IMA Journal of Numerical Analysis*, 17(2):305–320.

MANNI, C. 2001. On shape preserving C^2 Hermite interpolation. *BIT Numerical Mathematics*, 41(1):127–148.

MCALLISTER, D. F. AND ROULIER, J. A. 1981. An algorithm for computing a shape-preserving osculatory quadratic spline. *ACM Transactions on Mathematical Software (TOMS)*, 7(3):331–347.

MCALLISTER, D. F. AND ROULIER, J. A. 1981. Algorithm 574: Shape-Preserving Osculatory Quadratic Splines [E1, E2]. *ACM Transactions on Mathematical Software (TOMS)*, 7(3):384–386.

MOLER, C. B. 2008. *Numerical Computing with* MATLAB. The MathWorks Inc., Natick, Massachusetts.

PIAH, A. R. M. AND UNSWORTH, K. 2011. Improved sufficient conditions for monotonic piecewise rational quartic interpolation. *Sains Malaysiana*, 40(10):1173–1178.

PRUESS, S. 1993. Shape preserving C^2 cubic spline interpolation. *IMA Journal of Numerical Analysis*, 13(4):493–507.

QUINT, A. 2003. Scalable vector graphics. *IEEE MultiMedia*, 10(3):99–102.

RAMSAY, J. O. 1988. Monotone regression splines in action. *Statistical Science*, 3(4):425–441.

SCHMIDT, J. W. AND HESS, W. 1988. Positivity of cubic polynomials on intervals and positive spline interpolation. *BIT Numerical Mathematics*, 28(2):340–352.

SCHUMAKER, LARRY L. 1983. On shape preserving quadratic spline interpolation. *SIAM Journal on Numerical Analysis*, 20(4):854–864.

ULRICH, G. AND WATSON, L. T. 1994. Positivity conditions for quartic polynomials. *SIAM Journal on Scientific Computing*, 15(3):528–544.

WANG, Q. AND TAN, J. 2004. Rational quartic spline involving shape parameters. *Journal of Information and Computational Science*, 1(1):127–130.

YAO, J. AND NELSON, K. E. 2018. An unconditionally monotone C^2 quartic spline method with nonoscillation derivatives. *Advances in Pure Mathematics*, 8(LLNL-JRNL-742107).

ZHU, Y.,HAN, X. 2015. C^2 rational quartic interpolation spline with local shape preserving property. *Applied Mathematics Letters*, 46:57–63.

ZHU, Y.,HAN, X. 2015. Shape preserving C^2 rational quartic interpolation spline with two parameters. *International Journal of Computer Mathematics*, 92(10):2160–2177.