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 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 C^2 monotone interpolant. Algorithm and implementation details, complexity and sensitivity analyses, usage information, and a brief performance study 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, Documentation

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.

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 47405; `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 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 of quintic polynomials. The code by Fritsch [1982] for C^1 cubic spline interpolation is the predominantly utilized code for constructing monotone interpolants at present. Theory has been provided for the quintic case [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.

The importance of piecewise quintic interpolation over lower order approximations can be simply observed. In general, the order of a polynomial determines the number of function (and derivative) values it can interpolate, which in turn determines the growth rate of error away from interpolated values. C^2 quintic (order six) splines match the function value and two given derivatives at each breakpoint. This work provides a Fortran 2003 subroutine `MQSI` based on the 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 $Y_1 \le Y_2 \le \cdots \le Y_n$, find a C^2 monotone increasing quintic spline $Q(x)$ with the same breakpoints satisfying $Q(X_i) = Y_i$ for $1 \le i \le n$. (`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 an empirical performance study and some graphs of constructed interpolants.

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. In an attempt to capture the local shape of the data, 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 curvature.

Algorithm 1: `QUADRATIC_FACET`$\big(X(1{:}n),\, Y(1{:}n),\, i\big)$
where $X_j,\, Y_j \in \mathbb{R}$ for $j = 1, \ldots, n$, and $1 \le i \le n$. Returns the slope and curvature at X_i of the local quadratic interpolant with minimum magnitude curvature.

`if` $\big((Y_i \approx Y_{i-1})$ `or` $(Y_i \approx Y_{i+1})\big)$ `then; return` $(0, 0)$
`else if` $\big((Y_{i+1} - Y_i)(Y_i - Y_{i-1}) < 0\big)$ `then`
  The point $(X_i,\, Y_i)$ is an extreme point. The quadratic with minimum magnitude curvature 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` $\big(Df_1,\, D^2 f_1\big)$
  `else;` `return` $\big(Df_2,\, D^2 f_2\big)$
  `endif`
`else`
  The point $(X_i,\, Y_i)$ is in a monotone segment of data. In the following, it is possible that f_1, f_2, or f_3 does not exist because $i \in \{1, 2, n-1, n\}$. In those cases, the minimum magnitude curvature 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` $\big(|D^2 f_1| = \min\big\{|D^2 f_1|, |D^2 f_2|, |D^2 f_3|\big\}\big)$ `then; return` $\big(Df_1,\, D^2 f_1\big)$
  `else if` $\big(|D^2 f_2| = \min\big\{|D^2 f_1|, |D^2 f_2|, |D^2 f_3|\big\}\big)$ `then; return` $\big(Df_2,\, D^2 f_2\big)$
  `else;` `return` $\big(Df_3,\, D^2 f_3\big)$
  `endif`
`endif`

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`(x_0, x_1, f)
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 $[x_0, x_1]$.

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

The necessity of Steps 2 and 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 order six 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` $\big(60v - w\big(24Df(x_0) + 32Df(x_1) - 2t + w(3D^2f(x_0) - 5D^2f(x_1))\big) < 0\big)$
    `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` $\big(w\big(2\sqrt{Df(x_0)\,Df(x_1)} - 3(Df(x_0) + Df(x_1))\big) - 24v \leq 0\big)$
  `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. $\gamma := (4Df(x_0) - D^2f(x_0)w)\sqrt{Df(x_1)}/t$

$\beta := \dfrac{60v/w + 3\big(w(D^2 f(x_1) - D^2 f(x_0)) - 8(Df(x_0) + Df(x_1))\big)}{2\sqrt{Df(x_0)\,Df(x_1)}}$

`if` $(\beta \le 6)$ `then; return` $\big(\min\{\alpha,\gamma\} > -(\beta+2)/2\big)$

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

`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`.
`do` $i = 1, \ldots n$
  $\big(DQ(X_i),\, D^2Q(X_i)\big) :=$ `QUADRATIC_FACET`$(X,\, Y,\, i)$
`enddo`
Identify and store all intervals where Q is nonmonotone in a `queue`.
`do` $i = 1, \ldots n-1$
  `if not IS_MONOTONE`$(X_i,\, X_{i+1},\, Q)$ `then`
    Add interval $\big(X_i,\, X_{i+1}\big)$ to `queue`.
  `endif`
`enddo`
`do while` (`queue` of intervals is nonempty)
  Shrink (in magnitude) DQ and D^2Q that border intervals where Q is nonmonotone.
  Identify and store all remaining intervals where Q is nonmonotone in `queue`.
`enddo`
Construct and return a spline representation of $Q(x)$.

Since `IS_MONOTONE` handles both nondecreasing and nonincreasing simultaneously by taking into account the sign of v, 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 magnitude curvature 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, or intervals for which Q is nonmonotone at least once during the search. The binary search considers a Boolean function $B(s)$, for $0 \le s \le 1$, that is true if the order six polynomial on $[x_i, x_{i+1}]$ matching derivatives $f(x_i)$, $sDf(x_i)$, $sD^2f(x_i)$ at x_i, and derivatives $f(x_{i+1})$, $sDf(x_{i+1})$, $sD^2f(x_{i+1})$ at x_{i+1} is monotone, and false otherwise. As is outlined in Algorithm 3, the binary search is only applied at those breakpoints adjoining intervals for which Q is nonmonotone and hence $B(1)$ is false. It is further assumed that there exists $0 \le s^* \le 1$ such that $B(s)$ is true for $0 \le s \le s^*$ and false for $s > s^*$. 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 for all intervals. This multiple-interval binary search allows the value zero to be obtained for all derivative values in a fixed number of computations, hence has no effect on computational complexity. This binary search algorithm is outlined below.

Algorithm 4: `MULTIPLE_BINARY_SEARCH`$\big(X(1{:}n), Y(1{:}n), Q\big)$
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 $Q(X_i) = Y_i$. Modifies derivative values of Q at data points to ensure `IS_MONOTONE` is true for all intervals defined by adjacent data points, given a desired precision $\mu \in \mathbb{R}$.

Initialize the step size s, make a copy of Q, and construct three necessary queues for proceeding with the multiple-interval binary search.
$s := 1$
$\hat{Q} := Q$
`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*
`do` $i = 1, n-1$
  `if not IS_MONOTONE`$\big(X_i, X_{i+1}, Q\big)$ `then`
    Add interval starting at i to `shrinking`.
  `endif`
`enddo`
`do while` $\big($`searching or shrinking` is nonempty$\big)$
  Compute the *step size* for this iteration of the search.
  `if searching then;` $s := \max\{\mu, s/2\}$
  `else;` $s := 3s/2$
  `endif`
  `if` $(s = \mu)$ `then; searching := FALSE; endif`
  Increase DQ and D^2Q at all data points in `growing` that are strictly adjoining intervals for which Q is monotone.
  `do` $(i \in$ `growing`$)$ `and` $(i \notin$ `shrinking`$)$
    $DQ(X_i) := DQ(X_i) + sD\hat{Q}(X_i)$

$D^2Q(X_i) := D^2Q(X_i) + sD^2\hat{Q}(X_i)$

Add intervals starting at $i-1$ and i to `checking`.

`enddo`

Decrease DQ and D^2Q at all data points in `shrinking` and ensure those data point indices are placed into `growing`.

`do` $i \in$ `shrinking`

Add i to `growing` if it is not already present.

$DQ(X_i) := DQ(X_i) - sD\hat{Q}(X_i)$

$D^2Q(X_i) := D^2Q(X_i) - sD^2\hat{Q}(X_i)$

Add intervals starting at $i-1$ and i to `checking`.

`enddo`

Empty `shrinking` queue then check all intervals in `checking` for monotonicity with `IS_MONOTONE`, placing data points adjacent to intervals for which Q is nonmonotone into `shrinking`.

`do` $i \in$ `checking`

`if not IS_MONOTONE`$\big(X_i,\, X_{i+1},\, Q\big)$ `then`

Add i and $i+1$ to `shrinking`.

`endif`

`enddo`

`enddo`

In the provided package $\mu = 2^{-26}$, which results in a guaranteed 26 search steps for all intervals that are initially nonmonotone. An additional 43 steps could be required to reduce a derivative 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 the `MULTIPLE_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 `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. deBoor [1978] for the value, derivatives, and integral of a B-spline is also provided.

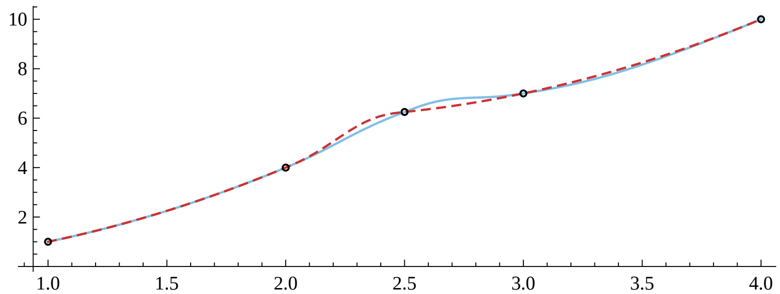

Fig. 1. A demonstration of the quadratic facet model's sensitivity to small data perturbations. This example is composed of two quadratic functions $f_1(x)=x^2$ over points $\{1, 2, 5/2\}$, and $f_2(x)=(x-2)^2+6$ over points $\{5/2, 3, 4\}$. Notably, $f_1(5/2)=f_2(5/2)$ and f_1, f_2 have the same curvature. 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.

4. COMPLEXITY AND SENSITIVITY

Algorithms 1, 2, and 4 have $\mathcal{O}(1)$ runtime. 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 generally provides aesthetically pleasing (low *wiggle*) initial estimates of the first and second derivatives at the breakpoints while perfectly capturing local low-order phenomena (e.g., a run of points on a straight line) in the data. The binary search for a point near the monotone boundary in $(\tau_1, \alpha, \beta, \gamma)$ space is preferred because it results in monotone quintic spline interpolants that are nearer to initial estimates than a policy that strictly shrinks derivative values.

5. PERFORMANCE AND APPLICATIONS

This section contains graphs of sample `MQSI` results given various data configurations. Computation times for various problem sizes are also provided. The files accompanying the subroutine `MQSI` offer multiple usages, namely `sample_main.f90` that demonstrates Fortran 2003 usage and a command line interface `cli.f90` that produces `MQSI` estimates for points in batches from data files. 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

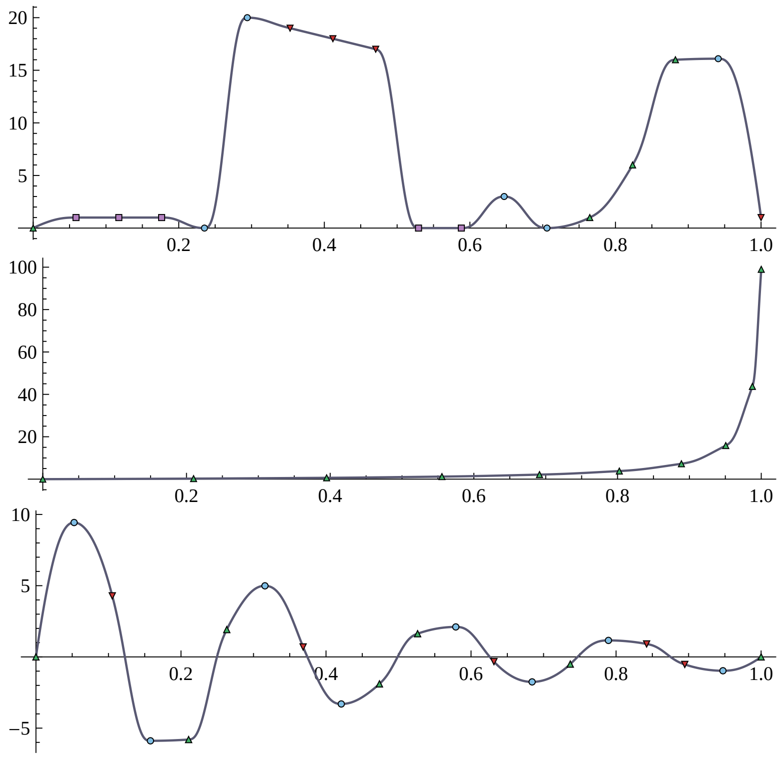

Fig. 2. `MQSI` results for three of the functions in the included test suite. The *piecewise polynomial* function (top) shows the interpolant capturing local linear segments, local flats, and alternating extreme points. The *large tangent* (middle) problem demonstrates outcomes on rapidly changing segments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

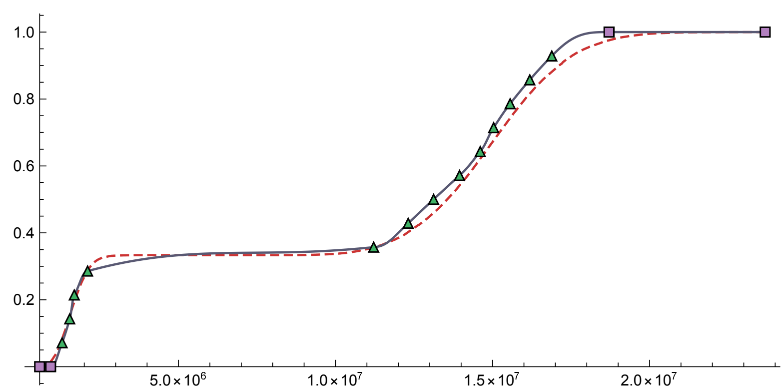

Fig. 3. `MQSI` results when approximating the cumulative distribution function of system throughput (bytes per second) data for a computer with a 3.2 GHz CPU performing file read operations from Cameron et al. [2019]. The empirical distribution of 30 thousand throughput values is shown in the red dashed 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.

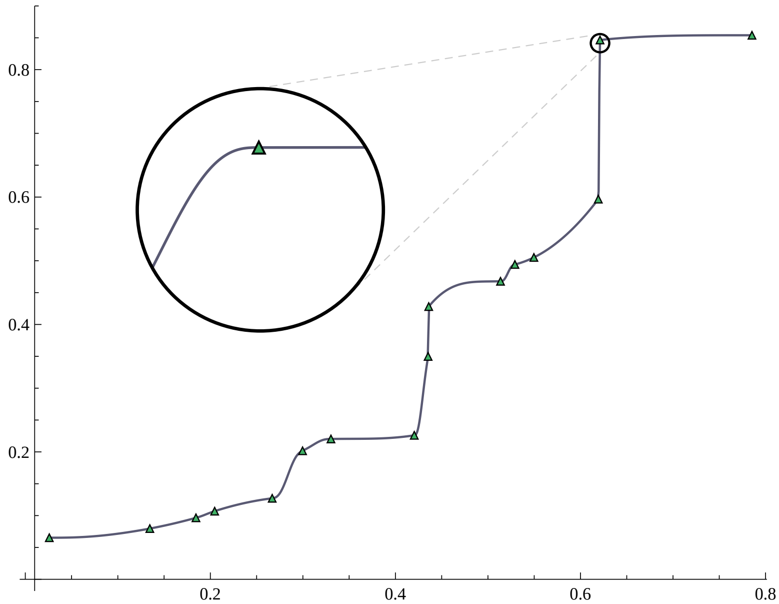

Fig. 4. The *random monotone* test poses a particularly challenging problem with large variations in slope. Notice that despite drastic shifts in slope, the resulting monotone quintic spline interpolant provides smooth and reasonable estimates to function values between data.

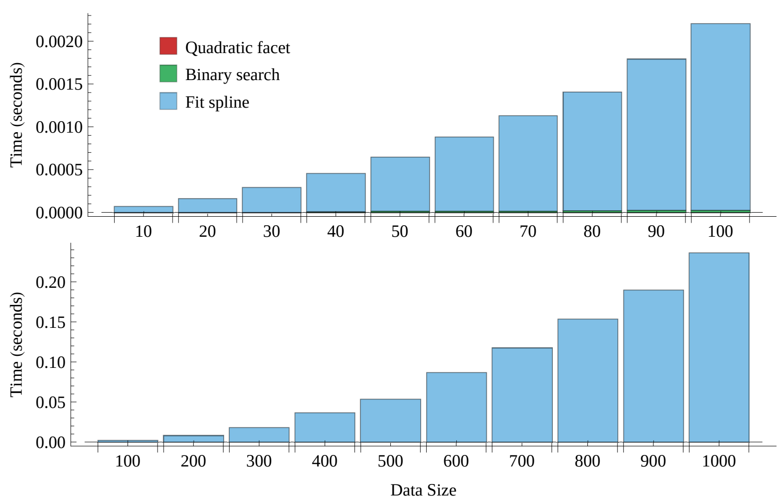

Fig. 5. Median total runtime of the routine `MQSI` for number of data points n=10 to 100 (top) and n=100 to 1000 (bottom), generated from 100 repeated trials averaged over 14 different testing functions. The timings are colored by major algorithmic component, but the vast majority of execution time is spent solving the banded linear system of equations in the `FIT_SPLINE` routine. The runtimes for the quadratic facet (Algorithm 1) take roughly one microsecond (10^{-6} seconds) per breakpoint, while the binary search (Algorithm 4) takes roughly four microseconds per breakpoint.

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.

Finally, Figure 5 provides execution times on a computer running MacOS 10.15.5 with a 2 GHz Intel Core I5 CPU. The total time is broken down by the major stages of `MQSI`. Most of the computation time is spent solving the banded linear system for B-spline coefficients in `FIT_SPLINE`.

BIBLIOGRAPHY

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.

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. 1985. Shape preserving spline interpolation. 19850020252, Brunel University.

Haralick, R. M., and Watson, L. T. 1981. A facet model for image data.. *Computer Graphics and Image Processing*, 15(2):113–129.

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

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.

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.

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

QUINT, A. 2003. Scalable vector graphics. *IEEE MultiMedia*, 10(3):99102.

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

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).