

Algorithm XXX: MQSI: Monotone Quintic Spline Interpolation

THOMAS C.H. LUX, LAYNE T. WATSON, TYLER H. CHANG,
WILLIAM THACKER, KIRK W. CAMERON, AND YILI HONG
Virginia Polytechnic Institute and State University

MQSI contains a serial code written in Fortran 1995 for constructing monotone quintic spline interpolants to 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. This paper includes algorithm and implementation details, complexity and sensitivity analyses, usage information, and a brief performance study.

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 [7, 12], aesthetic structural support in architecture [2], efficient aerodynamic surfaces in automotive and aerospace engineering [2], prolonged effective operation of electric motors [1], and accurate nonparametric approximations in statistics [9]. 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* [4, 5]. The specific shapes this work will achieve in approximations are

Authors' addresses: T. C. H. Lux, T. H. Chang, K. W. Cameron, Department of Computer Science, L. T. Watson, Departments of Computer Science, Mathematics, and Aerospace and Ocean Engineering, Y. Hong, Department of Statistics, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061; e-mail: tcflux@vt.edu.



Fig. 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.

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.

In statistics especially, the construction of a monotone interpolating spline that is C^2 continuous is meaningfully useful [13]. 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 [6], cubic [4], and (with limited application) quartic [16, 11, 18] cases. In addition, a statistical method for bootstrapping the construction of an arbitrarily smooth monotone fit exists [10], but the method does not take advantage of known analytic properties related to quintic polynomials. Theory has been provided for the quintic case [15, 8], 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 [17].

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 ‘fig:spline_demonstration’, 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 ‘`sec:monotone-cubic`’ summarizes the existing monotone cubic spline interpolation methodology, Section ‘`sec:monotone-quintic`’ presents an algorithm for constructing monotone quintic spline interpolants, Section ‘`sec:results`’ offers experiments and results with cubic and quintic methods, Sections ‘`sec:discussion`’ and ‘`sec:conclusion`’ discuss results and conclude.

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 [4] then expanded upon in [3]. Let $\pi : x_0 = k_1 < k_2 < \dots < k_n = x_1$ be a partition of the interval $[x_0, x_1]$. Let $f : \mathbb{R} \rightarrow \mathbb{R}$ 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 $f(k_i) \leq f(k_{i+1})$, $i = 1, \dots, n-1$, and $\Delta_i \geq 0$, $i = 1, \dots, n$, or $f(k_i) \geq f(k_{i+1})$, $i = 1, \dots, n-1$, and $\Delta_i \leq 0$, $i = 1, \dots, n$. Let \hat{f} be a piecewise cubic polynomial defined in each subinterval $I_i = [k_i, k_{i+1}]$ by

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

Notice that a trivially monotone spline results when $\Delta_i = 0$, for $i = 1, \dots, n$. However, such a spline has too many *wiggles* for most applications. [3] 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 $\alpha = (\Delta_i(k_{i+1} - k_i)) / (f(k_{i+1}) - f(k_i))$ and $\beta = (\Delta_{i+1}(k_{i+1} - k_i)) / (f(k_{i+1}) - f(k_i))$.



Fig. 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.

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 ‘fig:projection’.

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 ‘alg:check-monotone’ checks monotonicity, Algorithm ‘alg:make-monotone’ enforces monotonicity on an order six polynomial piece of the quintic spline, and Algorithm ‘alg:monotone-spline’ 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 , $g : S \rightarrow \{\text{FALSE}, \text{TRUE}\}$ is a right continuous Boolean function, $g(b) = \text{TRUE}$, and for $\mu \in [0, 1]$, $g((1-\mu)a + \mu b) = \text{TRUE} \implies g((1-\nu)a + \nu b) = \text{TRUE}$ for $\mu \leq \nu \leq 1$. The search returns $((1-c)a + cb)$ for the smallest $c \in [0, 1]$ such that $g((1-c)a + cb) = \text{TRUE}$.

These algorithms make use of first and second derivatives of the approximated function where D denotes the differentiation operator. 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`(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 ($f(x_0) = f(x_1)$) and not ($0 = Df(x_0) = Df(x_1) = D^2f(x_0) = D^2f(x_1)$)
- 2) return FALSE
- end if
- 3) if ($Df(x_0) < 0$ or $Df(x_1) < 0$)
- 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 [14], where α , β , γ , and δ are defined in terms of values and derivatives of f at x_0 and x_1 . Step ‘alg1-alpha’ checks for the necessary condition that $\alpha \geq 0$, Step ‘alg1-beta’ checks $\beta \geq \alpha$, and Step ‘alg1-gamma’ checks $\gamma \geq \delta$, all from [14]. If all necessary conditions are met, then the order six piece is monotone and Step ‘alg1-simplified-true’ concludes this check.

```

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

```

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

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

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

The variables A and B correspond directly to the theoretical foundation for positive quartic polynomials established in [15], first defined after Equation (18).

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

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

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

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

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

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

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 [15]. 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 ‘alg1-abg’ below.

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

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

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

$$\text{if } (\beta \leq 6) \text{ return } (\alpha > -(\beta + 2)/2)$$

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

$$\text{end if}$$

The reason for structuring the α , β , and γ computations in terms of the second derivative of f (seen in Step ‘alg1-abg’ of Algorithm ‘alg:check-monotone’) 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.

Notice that both Algorithms ‘alg:check-monotone’ and ‘alg:make-monotone’ 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 ‘alg:check-monotone’), while a constant set of operations and a single binary search are performed for enforcing monotonicity (Algorithm ‘alg:make-monotone’). 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 ‘alg:make-monotone’ only needs to recalculate Steps ‘alg1-abg’ and ‘alg1-last’ of Algorithm ‘alg:check-monotone’ during the binary search. Next, an algorithm for constructing a monotone quintic spline interpolant is presented.

Algorithm 2: `monotone_spline` $((k_1, \dots, k_n), f, s)$ where (k_1, \dots, k_n) is an increasing sequence of real numbers, f is an order six piecewise polynomial with breakpoints k_1, \dots, 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.

```

create queue
create changed(1, ..., n) := FALSE
create step_size(1, ..., n) := (Df(k1)/s, ..., Df(kn)/s)

```

The three variables defined in Step ‘step:create’ 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.

```

for  $i := 1, \dots, n - 1$  enqueue  $((k_i, k_{i+1}))$  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.

```

while (size(queue) > 0)
  ( $k_j, k_{j+1}$ ) := dequeue
  if (not is_monotone( $k_j, k_{j+1}, f$ ))
    if (changed( $j$ ))       $Df(k_j) := Df(k_j) - \text{step\_size}(j)$  if (changed( $j+1$ ))   $Df(k_{j+1}) :=$ 
     $Df(k_{j+1}) - \text{step\_size}(j+1)$ 

```

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.

```

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

```

Step ‘step:changed’ records the endpoints of the current interval as having been changed, while ‘step:requeue’ 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 [15] 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 ‘alg1-alpha’ through ‘alg1-gamma’ of Algorithm ‘alg:check-monotone’ to all fail, resulting in Step ‘alg1-simplified-true’ returning TRUE.

If successive monotonicity fixes are required for all intervals, the worst case runtime of Algorithm ‘alg:monotone-spline’ 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 ‘`alg:monotone-spline`’ 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 ‘`fig:cubic_quintic_sin`’. Given an increasing number of points, the maximum error of cubic and quintic approximations is shown in Figure ‘`fig:experiment_1`’. 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.

3.2 Random Number Generation

Consider the cumulative distribution function (CDF) defined by a mixture of three Gaussian distributions and visualized in Figure ‘`fig:experiment_2_distribution`’. 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 ‘`fig:experiment_2_results`’.

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 ‘`alg:check-monotone`’ and ‘`alg:make-monotone`’ 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 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.

BIBLIOGRAPHY

- [1] Tomas Berglund, Andrej Brodnik, Håkan Jonsson, Mats Staffanson, and Inge Soderkvist. Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles. *IEEE Transactions on Automation Science and Engineering*, 7(1):167–172, 2009.
- [2] AnnMarie Brennan. Measure, Modulation and Metadesign: NC Fabrication in Industrial Design and Architecture. *Journal of Design History*, 11 2019.
- [3] R. Carlson and F. Fritsch. Monotone piecewise bicubic interpolation. *SIAM Journal on Numerical Analysis*, 22(2):386–400, 1985.
- [4] F. Fritsch and R. Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

- [5] John A Gregory. Shape preserving spline interpolation. *Brunel University*, 1985.
- [6] Xuming He and Peide Shi. Monotone b-spline smoothing. *Journal of the American Statistical Association*, 93(442):643–650, 1998.
- [7] Daniel Lawrence Herman and Mark J Oftedal. Techniques and workflows for computer graphics animation system, December 2015. US Patent 9,216,351.
- [8] Walter Hess and Jochen W Schmidt. Positive quartic, monotone quintic c2-spline interpolation in one and two dimensions. *Journal of Computational and Applied Mathematics*, 55(1):51–67, 1994.
- [9] Gary D Knott. *Interpolating cubic splines*, volume 18. Springer Science & Business Media, 2012.
- [10] Florian Leitenstorfer and Gerhard Tutz. Generalized monotonic regression based on b-splines with an application to air pollution data. *Biostatistics*, 8(3):654–673, 2006.
- [11] Abd Rahni Mt Piah and Keith Unsworth. Improved sufficient conditions for monotonic piecewise rational quartic interpolation. *Sains Malaysiana*, 40(10):1173–1178, 2011.
- [12] A. Quint. Scalable vector graphics. *IEEE MultiMedia*, 10(3):99–102, July 2003.
- [13] James O Ramsay et al. Monotone regression splines in action. *Statistical Science*, 3(4):425–441, 1988.
- [14] Jochen W. Schmidt and Walter Heß. Positivity of cubic polynomials on intervals and positive spline interpolation. *BIT Numerical Mathematics*, 28(2):340–352, Jun 1988.
- [15] G. Ulrich and L. Watson. Positivity conditions for quartic polynomials. *SIAM Journal on Scientific Computing*, 15(3):528–544, 1994.
- [16] Qiang Wang and Jieqing Tan. Rational quartic spline involving shape parameters. *Journal of Information and Computational Science*, 1(1):127–130, 2004.
- [17] Yimeng Xie, Caleb B King, Yili Hong, and Qingyu Yang. Semiparametric models for accelerated destructive degradation test data analysis. *Technometrics*, 60(2):222–234, 2018.
- [18] Jin Yao and Karl E Nelson. An unconditionally monotone c2 quartic spline method with nonoscillation derivatives. *Advances in Pure Mathematics*, 8(LLNL-JRNL-742107), 2018.