Revision of TOMS-2021-0028

Dear reviewers,

We are deeply grateful for the comments, questions, and references you have all provided. In accordance with your feedback we have made significant changes to the manuscript to improve clarity, expand the summary of related literature, and incorporate visual comparisons with existing spline software. We hope you find these changes have resulted in an overall better manuscript.

Each numbered item and italicized text block below is a reviewer comment (or summary of similar comments). Our responses immediately follow.

(1) The authors should include references to more related works and justify the differences between this work and the many prior works on C^2 interpolation. [given 22 references]

In order to properly satisfy this request, we have significantly expanded the review of relevant research in the introduction (Section 1, paragraphs 3-6). This new content summarizes the individual tracks represented by the conglomerate of references provided. After reviewing all of those works we are still quite certain that this manuscript represents a novel and meaningful contribution to the literature, and have justified it's position among those works in the revised manuscript. The most outstanding contribution of this work remains the code, which applies sharp monotonicity constraints (unlike any other C^2 methods) to ensure a monotone quintic interpolant nearest to the initial nonmonotone (but otherwise *nice*) approximation is achievable.

(2) The authors should include references to and comparisons with the following software package (which appears to be very similar).

> *Costantini, P. An algorithm for computing shape-preserving interpolating splines of arbitrary degree, Journal of Computational and Applied Mathematics, 1998, 22(1), pp. 89-136.*

While the concerns about subjectivity in comparisons are acknowledged, users will still want to know the context of this work and the trade-offs would they use this package instead of other spline algorithms. For this reason a comparison with other methods including visuals should be provided.

We appreciate this perspective and have included such comparisons in the updated Section 5 (Figures 5-8). The comparison shows methods from four published spline software packages against MQSI for the same test problems as originally in the paper (and associated test code). An important point to note is that we were hesitant to include comparisons with the published R package for Schumaker spline interpolants, because (as is shown in the revised manuscript) the package often produces incorrectly nonmonotone approximations to data. In our opinion this undesirable outcome is a valid demonstration of the numerical difficulties that arise when writing codes for spline algorithms. They are not trivial to handle, and that underscores the importance of thoroughly and carefully peer reviewed implementations like what is provided by this work.

(3) The monotonicity of Q(x) is not proved.

Per Section 1 paragraph 5, Section 2 paragraphs 3, 4 and 6, and Algorithm 2 (all of the original submission) the theory proving the sharp necessary and sufficient conditions for monotonicity is provided by Ulrich and Watson [1994]. This code conforms precisely with the tight boundary conditions referenced therein. The theory is quite dense and it would seem inappropriate to reproduce all of that work in this manuscript as well. To further clarify that the conditions for monotonicity are proven, we have added an additional paragraph to Section 4 explaining this fact.

(4) The approximation order of the algorithm is not shown.

In the same paragraph of Section 4 referenced above, we have explained that the approximation order of the method is that of any second order approximation (because a local quadratic interpolant is used to approximate initial derivative values), but also note that the asymptotic approximation order is often of little importance in the context of (sparse) scattered data (the target application of most shape preserving techniques, especially those achieving higher levels of continuity).

(5) Why use piecewise degree 5? In principle, piecewise degree 3 is enough for C^2. The additional degrees of freedom for making the interpolant monotone could be obtained by adding knots between the breakpoints. It would be great if you could motivate your choice in more detail.

As stated in the comment, constructing polynomial pieces that interpolate arbitrary function values and two derivatives at the ends of an interval requires 6 degrees of freedom and hence an order 6 polynomial (degree 5, quintic) without adding additional knots. This work builds on theory that does *not* incorporate new knots, and it is our opinion that it is generally favorable to not insert new knots. Without the insertion of additional knots, the ability to construct lower order monotone C^2 splines is restricted to specific arrangements of data. The restriction of having only a fixed set of knots is somewhat arbitrary, but that is the application targeted by this line of research.

(6) Some statements depend on two variables being "approximately equal", but what does that mean? Equal up to some small threshold?

The definition of approximately equal here is when two floating point numbers are within the machine precision (at their scale) of each other. This is a rather standard approach for ensuring safe separation of floating point numbers. We have added a sentence explaining this to the pretexts of the algorithms that reference this operation.

(7) In Algorithm 1 (and also in the text), you use the term "curvature" to refer to the second derivative. I would avoid this, since "curvature" is a term reserved for curves, but you are dealing with functions. Similarly, the term "minimum curvature derivative" is inappropriate.

This has been rephrased throughout to reduce confusion.

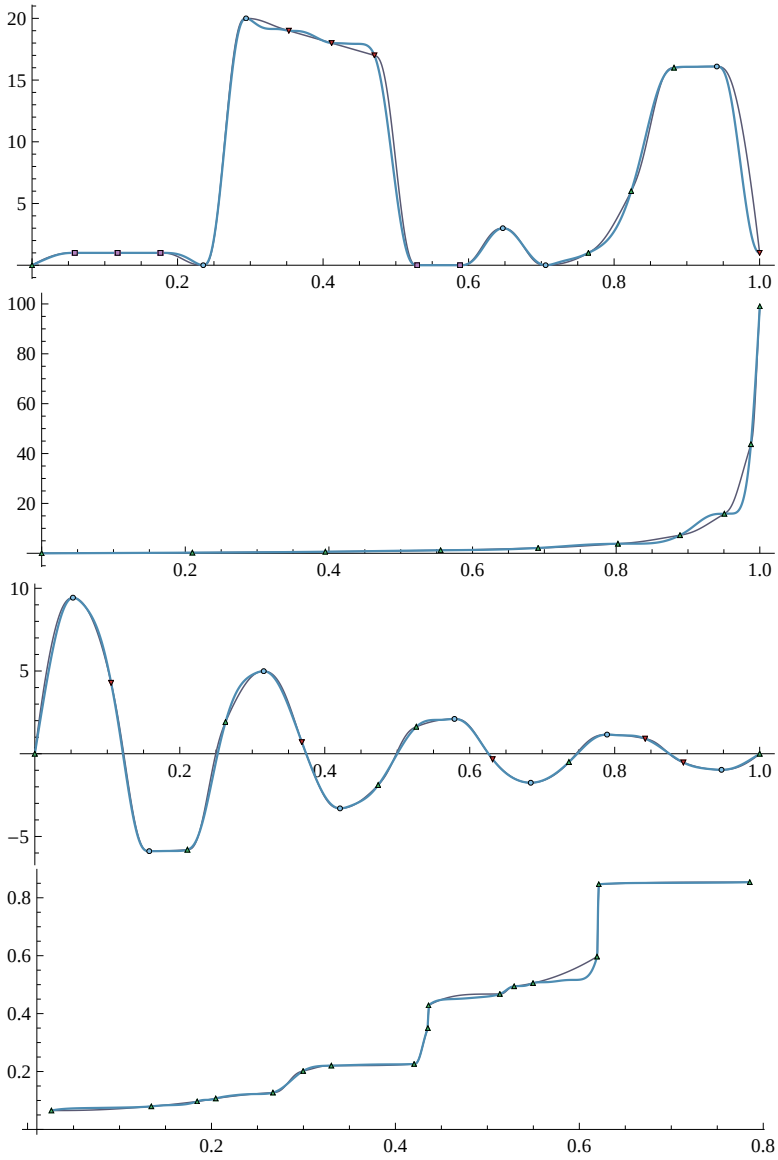
(8) In Algorithm 2, it might be better to say that f is a "degree 5" polynomial, instead of "order 6".

Noted and changed.

(9) Your binary search results in a particular set of derivatives, for which the local quintic piece is guaranteed to be monotone, but it is clearly not a unique choice. Alternatively, one could try to find, among all possible sets of derivatives and corresponding monotone quintic interpolants, the best, according to some criterion, e.g. least L2 norm. This would turn the problem into a constrained optimization problem, which is probably harder to solve, but it might, at the same time, give better results and overcome the limitation shown in Fig. 1. More motivating details would be appreciated.

This was a point of our own severe consideration before submitting the manuscript and this comment reflected our original thoughts. In due response to this we've implemented a naive L^2 minimization algorithm over the second derivative that uses gradient descent over the spline representation to enforce local monotonicity constraints in a Lagrangian fashion (with an adaptive multiplier on monotonicity violations). The results we achieved were quite mixed and we hesitate to incorporate commentary from this excursion into the manuscript. In truth, formally expanding on this trajectory would require an entirely new manuscript with deeper study of the problem, while it is not immediately obvious that the results would be *better* than the present approach. The global L^2 minimizer of the second derivative is simply the unique C^2 cubic spline

interpolant, but that is known to produce large and unnecessary local variations in the approximation (along with not being shape preserving). Included in this response (to reviewers only) are some of the preliminary visuals we produced in exploring this approach. This is by no means final and conclusive, but the results were mixed enough for us to warrant not including them in the paper.



L^2 Minimization Experiment. `MQSI` (light gray) compared with an experimental global L^2 minimization approach (blue) on each of the test problems from the paper. It is *very* difficult to consistently construct fits that lack local oscillations (i.e., minimize L^2) as well as maintain monotonicity with this approach. The method relies on the B-spline basis in this experiment, which affects the bias towards specific solutions when following the second derivative L^2 minimizing gradient. This work would warrant a significant line of research to properly resolve these issues in a theoretically sound way.

*(10) It is quite probable that a user would want to convert the piecewise Hermite representation of the interpolant to quintic B-spline form. But the authors say in Sections 3 and 4 that they solve a banded linear system of length n (n the number of data points) to make this conversion. However, this is not necessary. At each data point, only three of the quintic B-splines are non-zero. Therefore, the three associated B-spline coefficients can be computed from the value, first, and second derivatives of the interpolant at that single point. Thus one can just solve a small 3*3 linear system to get one group of 3 B-spline coefficients. Well, it requires a bit of algebraic work to write out this 3*3 linear system, but it can be done.*

As mentioned in the manuscript Section 5 paragraph 4, the banded linear system representation is favored for it's decreased redundancy. The computational complexity order is unchanged by solving a single banded system, while the global numerical stability is *theoretically* improved. The system of equations and number of nonzero splines at each breakpoint is thoroughly documented in `SPLINE.f90`, lines 172–206. In practice this approach is almost certainly slower than solving individual linear systems, but the increased global stability and decreased redundancy in representation is favored here (since the banded linear system also does not increase computational complexity). For users interested in constructing alternative representations, the optional argument `UV` documented in the code exists precisely for returning the computed first and second derivatives at knots.

(11) I'm confused by the O(1) and O(n) discussion in Section 4. If there are n data points then surely the main algorithm (estimating first and second derivatives at each data point) must require at least an O(n) algorithm, not O(1).

This was poorly worded on our part. The entire algorithm is O(n), however the binary search itself has a fixed number of loop iterations. The original intent of the statement was that the complexity order of the outer loop (the binary search) is constant, which ensures the overall complexity of O(n) is not changed. These sentences in Section 4 paragraph 1 have been rephrased.

(12) The output are values – why not the spline coefficients? If the output are values one would hope for a visualization package.

The code includes suitable documentation and examples to conform to any external visualization package desired by a user. Given the wide variety of mechanisms by which a user might want to visualize or use the results of this code, it is nearly impossible to provide a good generic solution to visualization.

For the convenience of reviewers, we are providing the code with which we generated the visuals for this manuscript. Although we note it has very specific system requirements to operate effectively. Namely: (1) running a POSIX shell (on macOS, Ubuntu, ...); (2) having `python3` of at least version 3.7 and the packages `numpy`, `tlux`, and `rpy2`; (3) having installed `gfortran` (or otherwise reconfiguring the codes to use another Fortran compiler of choice); and (4) having installed `R` (for Schumaker splines). The Fortran codes associated with this work as well as those from TOMS 574 (quadratic C^1) and BVSPIS from TOMS 770 are all wrapped to be callable directly from `python3`, which will be the parent process for all calls. A simple test usage can be seen in the provided `python3` file `test.py`. To be abundantly clear, this code is **not** intended for review and publication, but rather convenient access to various codes for reviewers to use for visualizing and comparing results. Precise installation command line instructions are provided at the end of this document.

(13) I was not familiar with the term/notion "quadratic facet model". The term facet made me believe that the paper was on bivariate interpolation. Nothing in the abstract or title prevents this misconception.

The term "univariate" has been added to the abstract for greater clarity on this point.

Throughout the manuscript, all modified text has been marked in red. We hope that you all find these revisions to have improved the manuscript, and we look forward to your response.

Sincerely,

Thomas Lux, on behalf of all authors.

Installing dependencies and running tests in `MQSI_testing.zip`

`Ubuntu 20+` command line instructions.

```
#() { echo "# $@" }
# install gfortran
sudo apt install gfortran
```

`macOS 12+` command line instructions (installs CommandLineTools and Homebrew first).

```
#() { echo "# $@" }
# install CommandLineTools if you do not have it already
xcode-select -p || xcode-select --install
# install Homebrew for package management
brew --version || /bin/bash -c "$(curl -fsSL \
     https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
# install gfortran
brew install gfortran
```

Common instructions: Python package installation, unzipping archive, and running `test.py`.

```
# install a visualization python package
python3 -m pip install --user tlux

# unzip archive, enter resulting directory, run tests
unzip MQSI_testing.zip
cd MQSI_testing
python3 test.py

# install R (if comparisons with Schumaker are desired)
# install Schumaker package through R graphical user interface
# run the following to install a Python package that wraps calls into R
python3 -m pip install --user rpy2

# make any desired modifications to test data at bottom of test.py
```