Editorial Office,
Journal of Computational Physics

February 14, 2016

Re: JCOMP-D-15-00846

Dear Editor,

We would like to appreciate the invaluable comments made by the reviewers. We have made adjustments to our article, titled '**Parallel level-set methods on adaptive tree-based grids**', to address the issues raised by the reviewers as detailed below.

- **Reviewer #1**
  The first reviewer has suggested to include "*some basic library codes in appendix, each matching its level set algorithm*". Indeed it is important for the scientific community to share knowledge and facilitate ways for reproducing scientific work for others. In the case of computational science, unfortunately, this is a very daunting task that is not always possible. Many scientific codes are highly specialized and are often too technical to be fitted into a paper.

  Specifically, in our case, our library amounts to nearly 30,000 lines of code, spread across more than 50 files and with complex dependencies. As such, it is practically not viable to include sample codes in the paper. Moreover, we believe inclusion of actual code, which involves many technical details, e.g. error checking, profiling tools, comments, etc., is not scientifically educational and may further alienate the reader from main message of the paper.

  Instead, and in accordance to the second reviewer who has raised related issues, we have opted to revise the pseudo-codes in the algorithms to make them more clear. We believe that the modifications should make the algorithms easier to understand and useful to the readers.

- **Reviewer #2**
  The second reviewer has raised several issues, mostly related to the pseudo-codes in the paper, as well as suggesting additional numerical tests. These issues are addressed as detailed below:

  - "*It took me a while to understand Algorithms 1 and 2. It is recom- mended that the authors pseudocode be closer to a language that many people are familiar with like C++ or Fortran 90 (or perhaps, the authors are using a new language that this referee is not aware of?).*"

    Our library is coded in '`c++`' (with a mix of little '`c`' code). We have made adjustments to the to algorithms to make them easier to understand. However, we believe it is best not to directly use a '`c++`' syntax to keep the pseudo-code language-agnostic and accessible to all readers.

  - "*In Algorithm 2, what is the definition of rank? owners rank? mpirank? st?*"

    There were multiple references to the "*rank*" in the pseudo-code that would have caused confusion. We have made things clear by explicitly referring to "*owners_rank*" and "*mpirank*". Also in the previous pseudo-code "*st*" was meant to indicate "`MPI_STATUS`". Again we have reorganized the pseudo-code to make this point more clear. We have also added more comments and text to the caption to explain things in more details.

  - "*In algorithm 2, if X is contained within the mesh on a given processor, does that mean all the points needed in order to derive the value at X are also on the same processor? What if the points in the interpolatory stencil of X belong to multiple processors? Which processor does the actual interpolation? Is it the processor that needs the value at X? or is it the processor that owns X?*"

    Since the Z-curve defines non-overlapping partition of the grid in parallel, it is guarantied that the rank found by the search operation in line 3 of algorithm 2, actually owns the interpolation point, i.e. there exists a *local cell* `c` belonging to process `owners_rank` which contains the interpolation point. This is because the search operation uses a binary search on the Z-curve to find the remote `owners_rank`.

    Our method for interpolation only requires that the values of the function (and optionally its second derivatives if using multi-quadratic interpolation) be defined on all vertices of cell `c`. This, in turn is guarantied by the fact that the boundary of each process is covered by a layer of ghost cells that have updated the ghost values at all their vertices before performing the interpolation. As a result, all the vertices of cell `c`, whether local or ghost, have valid values that can be used for interpolation.

– "*In section 4.2, the authors should do a standard 3D reversible test when checking speed-up. A standard test allows researchers to compare the new method to previous schemes. Also the authors should report the numerical error (e.g. the symmetric difference error or the level set error in proximity to the zero level set). For a reversible problem, the exact solution is known at the end time. So in Table 2, an extra column should be added to both the (a) and (b) parts which has the error (it is assumed that the error is independent of p?). For the reversible problem, the zero level set isosurface should be displayed at maximum deformation $t = T/2$ and at the end time $t = T$ for some representative values of CFL and lmax.*"

  The idea behind the test in section 4.2 was to look at the individual components that make the semi-Lagrangian algorithm and check their scaling and see how different parameters, most notably the CFL number, would affect the scaling results. In doing so we decided to look at the performance of a single iteration of semi-Lagrangian algorithm in isolation without including other ingredients, such as reinitialization, that would otherwise affect the timing and scaling. In fact the scaling results for a more comprehensive test are included toward the end of article and in the Stefan test.

  Also, our main focus in this article has been the scalability of several level set algorithm in parallel. The accuracy of many of these algorithms, albeit in serial, have been studied before in relevant articles that we have cited in details. However, we agree with the reviewer that it is important to show that the presented methods are in fact accurate and convergent. As a result we have added a new section, 3.4, that checks the accuracy of the standard rotation test as was suggested by the reviewer.

– "*Also in section 4.2, how frequently is reinitialization carried out? What is the pseudo time step $\Delta\tau$? What is τ_{stop}? If there is no reinitialization for this test, it is requested that the semi-Lagrangian algorithm together with reinitialization be analyzed in this section too*"

  Previous studies have shown that the adaptive time-stepping method that is used in this article greatly improves the convergence to steady state solution of the reinitialization equation (c.f. ref 33 in main text). We have generally observed that a few, typically 20, iterations of the algorithm is enough for the solution to converge within a small band (e.g. one smallest cell in the diagonal direction, i.e. $|\phi| < \sqrt{2}\Delta x_{min}$) around the interface. As a result we solve the reinitialization equation not to a final time (since each cell is using a different time-step) but to a fixed number of iterations. We have added a new paragraph in section 3.3 that clarifies this approach.

  As for the test in section 4.2, there is no reinitialization involved. This is because we are testing the effects of various parameters on the scalability of a single iteration of the semi-Lagrangian. We believe, however, that the reviewer is absolutely correct about the need to have a test of the entire algorithm. However, we argue that this role is already fulfilled by the Stefan test toward the end of the article.

The Stefan test of section 5 is a complete and non-trivial example and represents a typical scenario that one should expect when using our algorithms in practice. This example serves, not only as a demonstration of what our algorithms can achieve, but also as a scalability test of all different components, such as interpolation, semi-Lagrangian advection, reinitialization, solution of implicit linear system, and extension of this solution over the interface. It also compares these costs with costs associated with grid operations in its entirety which includes generation, refinement and coarsening, and partitioning of the forest. As figure 10, and table 7, illustrate, both the semi-Lagrangian and the reinitialization scale well in a real scenario @Arthur: How often do we call reinitialization and how many iterations each time? I think we should include these information. Thus we hope the results of this test is satisfactory enough for the reviewer.

@Frederic: What should we do about the scaling of the full rotation test? Should we do it? I have the feeling that the reviewer is probably not going to be satisfied ... Deadline is Feb 21

In light of these adjustments, we respectfully ask that our article be reconsidered for publication in the Journal of Computational Physics.

Sincerely,

Mohammad Mirzadeh (on behalf of all authors)