Reply to Anonymous Referee #3.

We would like to thank referee #3 for taking the time to review our paper. Our replies to the three main comments are below (in green).

1) There is no discussion of OpenMP as an alternative to MPI. Future hardware will require going to more shared memory and less message passing.

We acknowledge that there is no discussion of OpenMP in the current paper. OpenMP parallelization is not currently explicitly supported in Oasis3-MCT, and OASIS developers are aware of this shortcoming in the current implementation, especially in regards to possible future architectures. As indicated in the final section of the paper, OpenMP parallelization and performance of Oasis3-MCT on new architectures is something that is currently being explored by the development team. We hope to provide support in the next year (or so) and will have results at that time to share with the community.

2) There is no discussion of GPUs, MICs, etc and plans to port OASIS to novel architectures.

The OASIS development team is actively pursuing access and testing of OASIS on newer architectures and hope to have some results in the next year to share with the community. We recognize this is an important issue moving forward.

3) I am somewhat taken aback by the extreme cost of providing bfb (bit for bit) reproducing algorithms. In other similar codes this cost ratio is somewhat lower (which could of course mean that the non-reproducing modes in other codes are too slow!) This may require some work.

With regard to the cost of the bfb conservation computation, we were also quite shocked at the cost of the bfb operation. We have revised the discussion and results of the CONSERV transform in the paper significantly, adding global sum options that have been recently added to the OASIS infrastructure and that will be released in OASIS3-MCT_4.0. The OASIS3-MCT_3.0 timings showed a clear problem with the bfb CONSERV performance. OASIS3-MCT_4.0 will provide additional options, including an option called "reprosum" that produces bit-for-bit results on different core counts and decompositions while performing significantly better than the current "bfb" option. Please see revised section 2.4, 3.5, and table 3 in the paper.