Response to Referee Reports on

The Cost of Deterministic, Adaptive, Automatic Algorithms: Cones, Not Balls

by Clancy, Ding, Hamilton, Hickernell and Zhang

Thanks to the editor and two referees for their encouraging words and very helpful comments. Below is our response to all of you (merged together since some points were mentioned by more than one of you).

As an overall comment, we have tried to be consistent with the ideas of the information-based complexity (IBC), and define our terms and explain our results in a way that this community would understand. However, some of our ideas do not quite fit the classical IBC mold, and we have needed to stretch the mold, hopefully without breaking it. Moreover, we are also seeking to reach a broader audience, i.e., those who use automatic algorithms like MATLAB's `quad`, and expect them to work correctly. This is why we stress the need for rigorous theoretical guarantees, which might be taken for granted in the IBC world.

1. We have shortened the manuscript. The introduction has been shrunk. The tables and lists in the introduction have been deleted. Lemma 1, our first new result, is now on p. 7.

2. We have relegated the cost budget, $N_{\max}$, to a remark. It is not mentioned in the algorithms or theorems.

3. We have clarified the meaning of "automatic" to mean all algorithms that adjust the cost relative to the error tolerance given. All our full-fledged algorithms are automatic, including those that are not adaptive. Fixed-cost building block algorithms are used construct the automatic algorithms. "Adaptive" means that the algorithm adjusts its computational effort based on function data.

4. Certainly the title should not have referred to the "complexity of an algorithm". We have modified the title, but think it important to keep "adaptive" (a departure from many IBC papers), and "automatic" (a word used in the numerical analysis literature).

5. The definition of optimality has been clarified. There is only one type now. We now speak of optimal order, to indicate that the algorithms are optimal up to a leading constant.

6. The semi-normed linear space of input functions, $\mathcal{F}$, is not called a Banach space. The weaker semi-norm $|\cdot|_{\widetilde{\mathcal{F}}}$ is now truly weaker than $|\cdot|_{\mathcal{F}}$. We make no mention of a space $\widetilde{\mathcal{F}}$ ($\mathcal{G}$ in the old notation). The normed linear space of output functions is now called $\mathcal{G}$.

7. We have summarized the main assumptions that needed for our adaptive algorithms earlier, in Section 1.2.

8. We have altered our discussion of previous work on adaptive algorithms.

9. Our perspective may differ from one referee, who states "Adaption is not an advantage or an disadvantage per se, it is needed if [the set of interest] is a cone." If the set of input functions of interest is already fixed, then it is true that one only wants the best algorithm irrespective of whether it is adaptive.

But we would argue that adaption *does* have an advantage. Algorithms used in practice (and referenced in our manuscript) are adaptive. The reason they are is that this allows one to expend more computational effort for a harder problem and less effort for an easier problem. In fact, at the end of Section 4.2 we argue that our adaptive algorithms for cones enjoy a stronger sense of optimality than the non-adaptive ones do for the (convex set) ball. This point is mentioned again in Section 7.2. It has to do with our definition of the complexity of a problem.

Unfortunately, the adaptive algorithms used in practice have no guarantees, but they often do a great job. We in the numerical analysis and IBC communities should be working to find guarantees, and they will only be found if we move away from balls. We would argue that cones give us the best setting to have practical and theoretically justified adaptive algorithms.

For the reader who is only concerned with theoretically justified algorithms, guarantees are assumed, but for the larger audience of users of readily available adaptive, automatic software guarantees are not a given, but are sometimes implied. For example, the abstract of Battles and Trefethen (2004), which describes the Chebfun toolbox, reads "All functions live on $[-1, 1]$ and are represented by values at sufficiently many Chebyshev points for the polynomial interpolant to be *accurate to close to machine precision*." (emphasis ours). This statement is true for many functions, but not all, and there are no guarantees for which functions this statement must be true. It is for this audience that we use the word "guarantee" often in our paper.

10. The condition $\phi(c\boldsymbol{y}) = c\phi(\boldsymbol{y})$ is now only needed for the building block non-automatic algorithms described in Section 2.3.

11. One referee gives a simple proof that the cone is not convex, but it assumes that the cone contains a ball. None of our cones contain a ball, so this proof does not work. In fact, if the cone contained a ball, then it would contain all balls.

12. We have changed our notation so that $\mathcal{I} = \{N_1, N_2, \ldots\}$. We do not require that A_{N_i} be nested in $A_{N_{i+1}}$, so $\mathcal{I} = \{2, 3, \ldots\}$ is fine, but for every $n \in \mathcal{I}$ there needs to ba a next larger number $\tilde{n} \in \mathcal{I}$ such that A_n is nested in $A_{\tilde{n}}$. The n_i are now used exclusively for the elements of $\mathcal{I}$ chosen by the algorithms.

13. For the numerical experiments, the success rate is the percentage of cases for which the error tolerance is strictly met.

14. The referees pointed out many careless typos. Thank you so much. We have tried to catch them all.

Finally, let me stress that we appreciate the careful attention that the referees and the editor have given to our paper. The comments have inspired a number of changes that we hope have improved the paper a lot. We look forward to your feedback.

References

Battles Z, Trefethen LN (2004) An extension of MATLAB to continuous functions and operators. SISC 25:1743–1770