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. They have inspired a lot of thought about how we can clarify and improve our manuscript. Below is our response to all of the points raised.

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. 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 such as MATLAB's `quad`, and expect them to work correctly. This is why we stress the need for rigorous theoretical guarantees, which are not available for practical algorithms like `quad`, but which might be taken for granted in the IBC world.

1. We have shortened the manuscript. The introduction has been shrunk. The tables and list in the introduction have been deleted. Lemma 1, our first new result, is now on p. 7 rather than p. 11.

2. We have relegated the cost budget, $N_{\max}$, to a remark and to the numerical examples. 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 as well as the error tolerance.

4. Certainly the title should not refer 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 the article — 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 a disadvantage per se, it is needed if [the set of interest] is a cone." If the set of input

functions of interest has been fixed then it is true that one only wants to find the best algorithm irrespective of whether it is adaptive. But we would argue that adaption *does* offer 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. We define our cost in Section 2.2 to reflect that reality. All of our automatic algorithms that are defined over some nice subset, $\mathcal{N}$, of the normed space of input functions, $\mathcal{F}$, e.g., $\mathcal{N}$ may be the ball $\mathcal{B}_\sigma$ or the cone $\mathcal{C}_\tau$. We define the cost of the algorithm as the maximum cost over functions in $\mathcal{N} \cap \mathcal{B}_s$, where $0 < s < \infty$. Thus the cost depends both on ε and s. Adaptive algorithms defined on $\mathcal{N}$ may have a cost that decreases as s decreases, while non-adaptive algorithms have a fixed cost independent of s.

At the end of Section 4.2 we argue that our adaptive algorithms defined on cones enjoy a stronger sense of optimality because their cost decreases as s decreases. This is not the case for the traditional non-adaptive algorithms defined on the ball. This point is mentioned again in Section 7.2.

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

10. We use the word "guarantee" a lot in our article for a reason. For the reader who is only concerned with theoretically justified algorithms, guarantees are assumed. However, for users of readily available adaptive, automatic software, guarantees are usually not provided, but are sometimes incorrectly 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 that specify the conditions under which this statement is true. It is for the audience of practitioners that we use the word "guarantee" often.

11. 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.

12. One referee gives a simple proof that the cone is not convex, but the proof 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 the whole linear space.

13. 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 be 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.

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

15. The referees pointed out many careless typos on our part. Thank you so much. We have tried to catch them all.

Finally, we greatly appreciate the careful attention that the referees and the editor have given to our paper. The comments have led to a number of changes that we hope have improved the paper. 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