Author Response to Review of

A case for open communication of bugs in climate models, made with ICON version 2024.01

Ulrike Proske, Nils Brüggemann, Jan P. Gärtner, Helmuth Haak, Dian Putrasahan, Karl-Hermann Wieners, and Oliver Gutjahr
EGUsphere, doi:10.5194/egusphere-2024-3493

Combined response to reviewers 2 and 3

We thank the reviewers for their substantial feedback. We decided to respond in thematic blocks to the combined points from reviewers 2 and 3 to aid readability.

Misunderstandings

From our point of view, there have been some misunderstandings in the review that we would like to clarify. First, we were perceived as wanting to be "engaging journal format and peer-review workflows for documenting bugs". This is not what we wanted to argue for. As the last paragraph (l. 456 - 462) illustrates, using papers for bug documentation is just one idea that we only mentioned as a side note. However, we spent more effort on the idea to have an openly accessible bug database for each model (such as a development platform or `www.icon-model.org`), which seems to be the approach favored by the reviewers as well. Our purpose with the underlying paper was to supply Figure 1 as one example for how bugs could be communicated concisely. Where such figures should be published, is something that we left open deliberately, also since it most likely depends on the workflow of the different model developer communities.

Reviewer 2 suggested that "a public issue-tracking system with linked commit, discussion, release-notes entry and warning messages would better serve the purpose of openly communicating the bug." Our concern is that documenting the code well and openly is not enough to communicate bugs beyond the model development community. In particular, it does not deal with the implications that a bug has for potential previous work and even for future work. Note in this context that a bug in a climate simulation does not necessarily mean that the whole simulation is worthless. On the contrary, it might still be useful for many applications if the implications of a bug can be clearly identified. This holds in particular, but not only for studies that have already been published. Here, it is essential to obtain an idea as to the degree to which the results of those studies are still reliable. Therefore, the main purpose of our paper was to demonstrate an effective but not exhausting way to document physical implications of a given model bug. As Reviewer 2 asked: "How a user of the old code could tell from release notes if her/his usage is affected?" Our perception is that other scientists and end users are using the model or its output without following each development and code change in the past. Thus they need an accessible way to learn about and understand major bugs in the model, for which we wanted to give an example in the form of Figure 1. However, we acknowledge that one deficiency of our paper was maybe not making concrete suggestions where the bug discussion and Figure 1 could be made openly available so that model users become aware of it.

At the same time as giving this example with Figure 1, we were arguing the case for such communication. However, we understand from the reviews that this double purpose did not come across.

One minor point that we would like to clarify is the methodological question of Reviewer 3, regarding the bug's implementation history as detailed in Sec. 3.2: "Was the version control history not studied to see whether this was the case?" As stated in line 198, this is exactly how the material for Sec. 3.2.1 and 3.2.2 was collected.

We take what we perceive as misunderstandings as shortcomings in our communication and thank the reviewers for alerting us to them.

Points we gladly take up

The reviewers have raised helpful advice that we would gladly take up in a new version of the manuscript:

- We kept the `stress_ice_zero` flag in the code in order to be able to conduct sensitivity studies for this manuscript, but will remove the flag now that this is no longer necessary.
- Now that the manuscript has a DOI, we have linked it in issue 86 (the original report of the bug, `https://gitlab.dkrz.de/icon/icon-mpim/-/issues/86`).
- What the critique of missing testing procedures has made clear is that the manuscript needs some elaboration of existing testing procedures. Here, we want to emphasize that current ICON development is backed by both Continuous Integration tests for technical quality assurance and mandatory extended test suites for each change to the main release branch. The latter cover a number of European High Performance Computing environments and different compiler setups where available. They comprise regression tests running the full model setup for some period and comparing the results to a reference solution. The origin of "our" bug lies in the early model development where a "reference solution" would be a volatile target. The rigorous regression test was introduced only after that early development phase (even if long before the bug was identified). Nevertheless, the structure of some bugs, in particular when they are introduced in an early development stage of the model, prevents finding them until some particular incident, in this case a high-resolution simulation, brings them to developers' attention.
- As noted above, we appreciate the feedback from Reviewer 3 that a systematic approach to our study is lacking. It seems that in communicating and analysing one bug, detailing the hackathon experience, and making an argument for bug communication, we have perhaps overloaded the manuscript. In future efforts, we would condense the content or focus on one more systematically.
- We are sorry to have made so many oversights in porting the references from the reference manager to the paper format.

Conditions beyond our control

There are also some points in the review that concern the ICON development workflow and surrounding infrastructure. Here, structural changes are not straightforward to apply by us authors of this manuscript since we only reflect a small part of the entire community developing ICON. However, this does not mean that we do not attempt to improve the situation towards a more open and transparent environment. In fact, parts of the author team were heavily engaged in the transition from a proprietary to an open-source license of ICON. As correctly identified by Reviewer 2, the GitLab development is not open at the moment. However, the Max Planck Institute for Meteorology (MPI-M) has a roadmap for open development of its repository, which will be used to prototype such a development for the ICON consortium as a whole. In such an environment we believe that the approaches we are advocating will become even more important.

We would further add the remark that the software development culture at a research institute is certainly different from certain standards in the software development industry. Here, we believe that it is essential for us researchers to take over those standards that we can agree on to be beneficial but at the same time to maintain our freedom and effectiveness for future developments.

Understanding of climate model development and purpose

Finally, we think there are some points made in the reviews that point towards an under-appreciation of climate model purposes and development difficulties.

Climate model code is developed at the MPI-M for scientific reasons. Reviewer 3 asked "Is it possible that there are other instances of similar bugs in the codebase in which incompatible options can be enabled?" The clear answer is yes! Many switches are in the code to allow scientific sensitivity studies, and scientists are expected to choose them freely as fit to their question.

In addition, scientific code is not written to address specifications against which it can be tested. Often experimentation is needed to see the effect of a change. Then, if a bug is introduced with a new feature, subsequent regression testing does not help but serves to preserve the bug. The fact that in a research environment results are not known prior to investigation, poses fundamental problems for creating an adequate test environment. These are not replicated in a service environment, wherein code is developed and tested against known requirements.

Nevertheless, many of the testing mechanisms suggested by the reviewers are at least under way, as pointed out above. Current practices are expected to protect the code better against such mistakes, but there is no absolute guard. However, software practices at the time the bug was incurred were different, for example lacking thorough review and testing of commits. The task of working with and thoroughly testing enormous legacy code under the oracle and tolerance problem should not be underestimated, which we tried to make clear in the introduction.

The manuscript adds another point here, highlighting the bug's relationship to resolution: At a resolution of hundreds of kilometers, one is forced to work with and interpret statistics for which one has little intuition. Working at km-scale resolution, one can subject the results to physical thinking, and thus detect erroneous or unphysical behaviour, more easily.

The motivation for this manuscript comes out of many discrepancies noted by the reviewers. (ICON's) climate model development is not perfect in terms of workflows or product. The model is not service software but a scientific tool, and it will always be prone to contain bugs. Thus the question of how to "detect and prevent" them needs to be continuously reevaluated. However, given the assumption that there will always be bugs, we believe that an open communication of bugs along with their scientific consequences is of even higher importance. Because of how the code is used by non-developers, trying to parse the history to have a more effective way of communicating bugs (rather than digging through the git history and release notes) would be welcome. A demonstration of how such communication could look like was the main purpose of this manuscript.

This document was generated with a layout template provided by Martin Schrön (`github.com/mschroen/review_response_letter`).