Reviewer 3

The paper is about a re-implementation of the ICON dynamical core using a domain-specific language embedded into Python called GT4Py. The work is carried out in the EXCLAIM project for which the paper presents the outcomes of the first phase. Described is the porting approach when rewriting the dynamical core into GT4Py, the testing strategy during the work, an evaluation of the computational performance and the scientific validation of the new code.

I found the paper was written in an accessible way, with a clear and sensible structure that covers all relevant angles of this development. The achieved milestone of the dynamical core rewritten in GT4Py is a remarkable achievement and the approach that utilised a very thorough testing procedure was well designed to avoid mistakes as much as possible. I would recommend a few minor edits to improve the overall presentation, which I list below with reference to the relevant sections of the text:

We thank the reviewer for taking time to reviewe the paper and for appreciating our work.

The abstract presents a specific throughput number but without specifying for what configuration or resolution. I would either add more details or leave it at the statement that the GT4pPy core exceeds ICON OpenACC performance without giving a specific number.

Thanks for point it out. The abstract has been accordingly adjusted.

The overview of current performance numbers in the paragraph in ll. 58ff is a wild mixture of very different configurations and resolutions. The intention is likely to take stock of how close current ESMs get to the 1 SYPD target, but this gets lost in the presentation. I would suggest to make this a little more focussed, ideally using a more like-for-like comparison. Moreover, most numbers are presented without references (NICAM, IFS-FESOM, ICON@1.25km). Some should stem from the GB submissions (https://dl.acm.org/doi/10.1145/3712285.3771789 and https://dl.acm.org/doi/10.1145/3712285.3771790) but it is irritating to see them published in this preprint before the availability of the original papers, particularly when no reference is given.

Thanks, again. The paragraph has been adjusted. Please see lines 62-70 of the revised manuscript.

In l. 97ff the three-phase nature of EXCLAIM is mentioned but no further information about the planned content of phases 2 and 3 is provided. Does this correspond to the deliverables shown in Figure? In the same paragraph, it is stated that the rewrite is "driven by the existing Fortran driver", which I did not understand until much later. Maybe this could be described in a form that makes it clearer that it is embedded into the existing Fortran framework, replacing calls to the dynamical core routines.

Done. Please see lines 97-100 in the revised manuscript.

Figure 1 is a useful illustration of the GT4Py code generation pipeline. I suspect not every reader may be familiar with therein used acronyms "GTIR" and "GTFN", which could be spilled out in the caption. GTIR is clarified later in the text but GTFN remains unclear.

Thanks for poiting it out. Figure 1 caption has been modified to explain the acronyms.

In l. 154f, three execution modes for running GT4Py are mentioned. Which of these are used here? Given that this is embedded into Fortran, I suspect this requires AOT?

Yes. A sentence has been added in the revised text to make it clear.

I did not immediately recognize the term "Fortran+" in l. 169 as the introduction of nomenclature. Maybe putting this in quotes would be helpful?

Done.

The description of the refactoring work in Sec. 4 is well written with an appropriate level of detail. The formatting of Listing 2 is unfortunate, with a page break between the listing and the caption - this should be rectified before final publication.

I agree on the readability angle in l. 278 but I did not understand the reason why only Python should allow in-line documentation through docstrings. I would argue that this could be done in any language, including Fortran.

You are right. We have modified the sentence to indicate that it *easier* to do so in Python.

The resolution of Fig. 4 seems a little low, it shows some artifacts in my print-out.

Thanks, again. Figure has been updated.

The hierarchy of testing levels appears well thought-out and seems effective to cover testing from a fine-grained stencil-loop level to full system regression. How much of this is automatic and when is it run? How expensive are these tests (in core-h or similar)?

Testings in level 1 and 2 are automatic and are part of the CI. For smaller experiments, it takes about 90 seconds for level 1 and about 30 seconds for level 2 for each experiment. Level 3 is full scientific simulation, which takes long and is only performed during this development phase. We do not expect to maintain it once the development is over.

The presented performance numbers are promising. However, in Fig. 7 either the plot colours or caption are wrong. The caption claims "GT4Py (dashed yellow) is about 10% faster than the Fortran+ (solid yellow)", while the plot suggests this to be the other way round. For the blue colours, dashed/solid seem to be reversed, so I suspect this may simply be a mistake in the caption.

There was indeed a mistake in the caption. Thanks for spotting it.

Given the substantial performance speed-up claims from the speed-of-light implementation: is there a specific pattern/generic improvement that accounts for this improvement? Or is it a mixture of several different changes?

In SOL implementation they used several tricks in addition to writing the code in CUDA C++ and inlining/fusion. The overall gain seem to be a result of all these combinations.

Since I'm not an expert on the scientific evaluation presented in Sec. 6, I cannot give a substantial feedback to this part.