Response to the Referee Comments

Manuscript: egusphere-2023-2129
Title: A dynamic approach to three-dimensional radiative transfer in subkilometer-scale numerical weather prediction models: the dynamic TenStream solver v1.0
Authors: Richard Maier, Fabian Jakub, Claudia Emde, Mihail Manev, Aiko Voigt, and Bernhard Mayer

Response to Referee #1

We thank Peter Ukkonen for his comments on our manuscript, which we will respond to below. To structure our response, his comments are printed on a gray background color, while our answers are displayed on ordinary white background.

Could you try to measure the number of floating point operations? This would give a direct measure of the cost. Reporting just the speed against the ten-stream implementation in Libradtran (a non operational code) is not that informative. Whether dynamic TenStream becomes a viable contender for NWP models will largely depend on how fast it eventually is, which depends both on the cost (number of FLOPs) and whether it can effectively exploit hardware (FLOPs per second). I agree with leaving this second aspect for a future paper to address but the number of floating point operations could easily be measured with GPTL (https://github.com/jmrosinski/GPTL) or other timing library built with PAPI.

By using GPTL, we were indeed able to quantify the number of floating point operations for all the solvers we use when applied to the very first time step of our shallow cumulus time series – except for the MYSTIC solver, whose timing would take a very long time:

	solar spectral range	**thermal spectral range**
δ-Eddington *1D two-stream solver*	$4.66 \cdot 10^{10}$ FLOPS (x 1.00)	$6.77 \cdot 10^{10}$ FLOPS (x 1.00)
dynamic TenStream *incomplete 3D solver with two Gauß-Seidel iterations*	$2.20 \cdot 10^{11}$ FLOPS (x 4.72)	$2.78 \cdot 10^{11}$ FLOPS (x 4.11)
original TenStream *full 3D solver*	$1.79 \cdot 10^{12}$ FLOPS (x 38.44)	$1.11 \cdot 10^{12}$ FLOPS (x 16.46)

However, we have decided to not include these numbers into the paper itself as GPTL printed out a few error messages during these tests that we were not able to get rid of. Hence, we are not completely sure how reliable these numbers are, although they in general confirm the relative speed numbers reported in Table 1 of the paper.

Besides that, we think that the number of floating-point operations is not an ideal measure of speed anyway. One of the primary computational expenditures in the (dynamic) TenStream solver for example is the retrieval of the TenStream coefficients from the corresponding look-up tables, which is mainly limited by memory bandwidth. On the other hand, calculating the Eddington coefficients in the δ-Eddington approximation is definitely faster, but features a significant amount of floating-point operations, indicating that even the number of floating-point operations is not an objective measure of performance.

So as much as we understand the desire to have an objective measure of how fast our dynamic TenStream solver actually is, providing such a number is a difficult task. And as we already noted in

the initial reply to your review, the main aim of this manuscript is to demonstrate the feasibility of the concepts behind the dynamic TenStream solver and not to compare its performance to operational radiation codes. The only statement we wanted to make in terms of speed is that a solver using incomplete solves is pretty fast by its design, as it just performs a fraction of the computations a full solve does.

Regarding the second aspect, I am in fact a bit concerned that the Gauss Seidel implementation, even if divided into subdomains, won't be able to exploit SIMD vectorization on CPU's. (Perhaps GPU's could be a better fit). You need not address this in the paper but do you think it would be a lot of work to write a Jacobi implementation for dynamic TenStream in the future in order to explore which gives better speed/accuracy trade-off on different hardware? I understand its convergence speed is worse but it could turn out to be a reasonable trade-off if it allows SIMD vectorization and Gauss-Seidel doesn't, especially as CPU hardware is moving towards longer vector lengths with AVX-512 instructions being supported by newer CPU's.

Implementing the Jacobi method into the dynamic TenStream solver is actually a pretty straightforward task and has already been done (but is not used by default).

Multicolor or red-black Gauß-Seidel/SOR solvers allow to use shared memory parallelization and SIMD instructions and should also allow for reasonable vector lengths for GPU processing, albeit again reducing convergence speed. We agree that if a host model is targeting accelerators or vector machines one should investigate the respective performance.

Line 66. SPARTACUS was not designed for NWP specifically, in the original Hogan and Shonk (2013) paper the authors actually talk more about climate models. The proliferation of (sub-) kilometer-scale NWP models arguably makes SPARTACUS more relevant for climate rather than NWP.

Thanks for pointing this out. By quantifying sub-grid scale 3D effects, SPARTACUS is indeed also relevant for climate models. But since especially global-scale NWP models still operate at resolutions where individual grid boxes contain both cloudy and cloud-free regions, these sub-grid scale 3D effects may also still play a role at the resolutions of currently employed NWP models. Hence we decided to change the expression "specifically designed for the use in NWP models" to "specifically designed for the use in large-scale models" for the second revision of the manuscript, as it was used in the introduction of the original SPARTACUS paper (Schäfer and Hogan, 2016). However, this change is no longer relevant due to the response to the following comment.

Line 67. This speed comparison is slightly out of date since SPARTACUS was recently sped-up by ~3x via code optimization. A better comparison might be to TripleClouds, it's fully 1D-counterpart, and to say it's 3-5 times more slower than TripleClouds (citing Fig. 3 in Ukkonen and Hogan, 2024). However, this is nitpicking sightly and it's also OK to leave the older comparison (optimized SPARTACUS against an optimized McICA could still still be ~6x slower for all I know).

References:

Hogan, R. J., & Shonk, J. K. (2013). Incorporating the effects of 3D radiative transfer in the presence of clouds into two-stream multilayer radiation schemes. Journal of the Atmospheric Sciences, 70(2), 708-724.

Ukkonen, P., & Hogan, R. J. (2024). Twelve Times Faster yet Accurate: A New State-Of-The-Art in Radiation Schemes via Performance and Spectral Optimization. Journal of Advances in Modeling Earth Systems, 16(1), e2023MS003932.

Thanks for pointing out this new, very interesting paper. In that regard, the speed comparison is indeed a bit outdated. Since we primarily wanted to refer to the currently relatively slow speed of

inter-column 3D radiative transfer solvers, we decided to get rid of this comparison for the second revision of the paper, because SPARTACUS is addressing sub-grid scale rather than inter-column 3D radiative effects and its significant speed-up is making the comparison pointless.

Response to Anonymous Referee #3

We thank Anonymous Referee #3 for his or her comments on our manuscript, which we will respond to below. To structure our response, the referee's comments are printed on a gray background color, while our answers are displayed on ordinary white background.

I am satisfied with the authors' responses to my previous review. I feel that they have made a real effort to understand my questions and to take into account my suggestions when relevant (as well as those from other reviewers). In addition to minor changes, the revised version of the manuscript includes a new section with an analysis of the behaviour of the scheme as a function of the number of Gauss-Seidel iterations; a welcome addition! Some of my questions were not answered in the manuscript because the authors considered them out of scope, which I found very reasonable. I can only hope that their next papers will investigate these compensating errors in the thermal, or the idea of advecting flux fields along with the other meteorological fields, or accounting for subgrid heterogeneity... In the meantime, I strongly recommend publication - after the four remaining occurences of "almost perfectly" are removed!

Thank you for your positive response on the revised version of our manuscript. For the second revision, we have also removed the four remaining occurrences of "almost perfectly". In particular, the following changes were made:

The sentence starting in l. 660 of the revised version was changed to: "At first glance, we can see that the results for the new solver are very similar to those obtained by the original TenStream solver in panel (b), even when operated at the low calling frequency of 60 s."

In l. 703, we modified the corresponding sentence this way: "In summary, we can hence say that for both the solar and the thermal spectral range, dynamic TenStream is able to visually almost reproduce the results obtained by the original TenStream solver, even when operated at lower calling frequencies."

The sentence starting in l. 739 was changed to: "In terms of heating rates, we saw that our new solver is almost able to reproduce the results of the original TenStream solver, even when operated at lower calling frequencies."

And finally, we also modified the sentence starting in l. 762 to avoid the phrase "almost perfectly": "de Mourgues et al. (2023) for example showed that in the thermal spectral range, even 30 quadrature points are sufficient to calculate heating rates that are very similar to those obtained by a line-by-line calculation."

References:

- Schäfer, S. A. K., Hogan, R. J., Klinger, C., Chiu, J. C., and Mayer, B.: Representing 3-D cloud radiation effects in two-stream schemes: 1. Longwave considerations and effective cloud edge length, Journal of Geophysical Research: Atmospheres, 121, 8567–8582, https://doi.org/10.1002/2016jd024876, 2016.