We would like to thank the referees for their careful reading of our manuscript. We are most grateful for the comments, constructive criticism and very useful suggestions received on how to improve the paper. Please find attached our detailed answers to the questions and a new version of the paper, which we hope satisfactorily to address the points raised during the discussion. The referee's comments are presented in black, and our responses in blue.

REFEREE 1 COMMENTS:

- Consider adding how the chemistry-kernel speedup translates into whole-model wall-clock savings

We thank the reviewer for this suggestion, as the system-level impact of the FP32 optimization is of interest for Earth system modellers. To address this, we have extracted the total wall-clock time for the full production EMAC simulations and split into the constant non-chemistry component and the chemical kinetics integration time. Figure 11 was added to visualize the whole-model wall-clock saving for both T42 and T106 resolutions as a function of the number of GPUs used.

The following discussion was added in Section 3.2.1 (line 323):
"To assess the GPU performance gains for the full EMAC model, we evaluated the total simulation wall-clock time. We note that there already a benefit of a factor of times 1.75 speedup in DP (Alvanos and Christoudias, 2019), and with the additional 30% speedup in SP, chemical kinetics account for only 17.7% of the total runtime at T42 resolution and 11.3% at T106 – from up to 90% on CPU-only configurations (Christou et al., 2016). The approximate 30% performance gain in SP from MEDINA yields substantial reduction in the total time-to-solution of 252/25.4 to 9.8/6.2 minutes in T106/T42 respectively. Fig. 11 shows the total wall-clock time in minutes for one-month simulation in T42 and T106 resolutions, separating the time spent in the accelerated chemistry solver from the rest of the EMAC runtime. For T42 resolution, the employment of Single precision saves an additional 25.4 minutes of total execution time on 4 GPUs, and maintains proportionally higher time saving with more nodes, as the rest of the model scales on the CPU. In the computationally more demanding T106 resolution, over 4 hours (250 minutes) are saved off the total simulation wall-clock time for 4 GPUs. As the model scales on multiple nodes, the 30% gain in chemical kinetics compared to DP is maintained, resulting in significant speedups with the scalability for the rest of the EMAC components on CPU. A detailed analysis of the substantial speedup already achieved in DP is presented in (Alvanos and Christoudias, 2019). For chemistry

schemes that require even larger fraction of the total EMAC runtime, for instance in Pozzer et al. (2022), the overall wall-clock improvement is expected to be even larger."

- The comparison between CPU-side and GPU-side typecasting (Sec. 2.1) is stated only qualitatively ("extensively testing… found to require overall more execution time"). Please consider adding quantitative numbers.

We agree with the reviewer that quantitative results clarify and strengthen this argument. The following sentence was added in line 152:

"…resulting in less than 1% overall acceleration for the chemistry module when compared to pure Double Precision…"

- Line 22: Please consider explaining why adaptive time-step control is relevant enough to justify the phrase "the numerical integration of the stiff chemical ODEs remains computationally demanding despite recent optimization of the adaptive time-step control."

We thank the reviewer for requesting this clarification. We have expanded the introduction to detail the impact of recent adaptive time-step control optimizations.

The following paragraph was added (line 24):

"A common ESM bottleneck is the simulation of atmospheric chemical kinetics (Christou et al., 2016). The numerical integration of the stiff chemical ordinary differential equations (ODEs) remains computationally demanding despite recent optimization of the adaptive time-step control (Dreger et al., 2025). While the number of function evaluations was reduced by up 25 to 43% for gas-phase chemistry, the overall computational time reduction was lower, at around 11%, while also requiring some adjustments to the original code"

- Since floating-point error can accumulate nonlinearly over longer integrations, consider discussing whether the reported <0.004% mass differences are expected to remain stable over longer runs, or over time during the one-month simulation.

We thank the reviewer for raising this important point regarding the long-term accumulation of floating-point errors. We have added a discussion to the manuscript to address this, noting that while double-precision has been proven stable over year-long runs, empirical validation of single-precision over similar timescales is deferred to future work due to computational constraints and the recent decommissioning of our test cluster.

The following paragraph has been added in line 284 of the manuscript:
As demonstrated in previous studies, GPU-accelerated double precision chemistry remains numerically stable over year-long integrations (Alvanos and Christoudias, 2019). The same behaviour is expected for the single-precision implementation due to the tightly constrained error profile and the consistent mass conservation observed across the multiple diurnal cycles within the one-month simulations that have been extensively tested for the purpose of this research. Due to the significant computational demands, the extended required simulation time for year-long runs, and the recent decommissioning of the HPC cluster that was used for this research, the experimental verification of this argument is deferred to future work.

- Line 135: Consider adding the comparison of the memory footprint against approach 1, for example: requires ~1.5× the memory footprint of a double-precision-only GPU kernel on the device (~3× that of Approach 1's device allocation, since Approach 1 only stores the single-precision array on-device).

We thank the reviewer for this helpful suggestion. We agree that explicitly outlining the memory footprint differences between the two approaches provides valuable context for the reader. We have contacted an experimental verification of this argument using the box model configuration and quantified the absolute memory requirements for Double Precision and the two Single Precision Approaches.

The following text has been added in line 141:

“Indicatively, experimenting with the box model configuration on a vector length (V L_GLO) of 57600, showed for the baseline Double Precision simulation 958MBs of Device Memory requirement. For Single Precision with host-side typecasting the requirement nearly halves to 490MBs, whereas on device-casting Single Precision the footprint is increased to 1446MBs.”

- Figure 2 compares FP32-GPU against FP64-CPU, which conflates precision differences with CPU/GPU architectural differences. The authors do isolate architectural noise via an FP64-CPU-vs-FP64-GPU control run (L178-179, ≤0.0004%) and do provide a same-architecture FP32-GPU vs. FP64-GPU comparison for the full EMAC model (Fig. 5), but not for the box-model case shown in Fig. 2. For consistency with Sec. 3.2 and to fully isolate the precision effect in the box-model results as well, consider adding a box-model panel analogous to Fig. 5 (FP32-GPU vs. FP64-GPU), or at minimum reporting the FP64-CPU-vs-FP64-GPU control run visually alongside Fig. 2.

Following this suggestion, we have expanded Fig. 2 into a two-panel figure that now includes a direct, identical-architecture comparison of the FP32 GPU implementation against the FP64 GPU solution.

We revised the text in 3.1 to read:

“Fig. 2 shows on logarithmic axes the final concentrations of all species as output by single precision (FP32) GPU-accelerated code against the double precision (FP64) GPU (left) and CPU reference (right). The points align along the 1:1 line across the full range spanning fifty orders of magnitude, indicating excellent agreement for both the configurations.”

- Table 2 reports speedup as a percentage using S[%] = (t_DP − t_SP)/t_SP × 100 (Eq. 1), normalized by the new (SP) runtime rather than the baseline (DP) runtime — a valid but less standard convention that risks being misread as (t_DP − t_SP)/t_DP. For consistency with the "Speedup ×" notation used in Fig. 8 (Eq. 4), and for more intuitive interpretation, consider reporting Table 2 as a dimensionless ratio (t_DP/t_SP) instead of a percentage. Note this would not make the two speedup metrics identical — Table 2 compares SP vs. DP at fixed solver/resolution, while Fig. 8's Si is normalized to a fixed single-node DP baseline for scaling — but using the same "×" format throughout would avoid the current mix of two different speedup conventions (Eq. 1 vs. Eq. 4) in the same paper.

Following the reviewer’s suggestion, we have updated Equation 2 and Table 2 to report the dimensionless ratio (t_DP/t_SP).

The corresponding text in Section 3.1 (line 242) has also been revised:

“Profiling has also revealed improved kernel execution time for the Single Precision implementation compared to Double, with a speedup (Eqn. 4) of 1.42×–1.45× depending on the selected solver (Table 2).”

- Table 2 shows a speedup of only ~42–45% from FP64 to FP32, well below the ~2× that might naively be expected either from halving data volume (memory-bound case) or from the GPU's ~2× FP32:FP64 FLOP throughput ratio (Table 1: 19.5 vs. 9.7 TFLOPS). Table 3 shows the memory-traffic reduction itself is non-uniform across the cache hierarchy (20.75% at L1-L2, 63.24% at L2-DRAM), suggesting the kernel's speedup is not simply explained by a uniform halving of memory traffic. Since the Introduction motivates FP32 partly via the Roofline model (Williams et al., 2009), consider adding a brief discussion — ideally informed by a roofline-style analysis or profiling breakdown — of whether the Rosenbrock integration kernel is memory- or

compute-bound, and whether non-floating-point operations (integer indexing, pivoting, adaptive step-size control/branching) that don't benefit from reduced precision are capping the achievable speedup below the theoretical 2×.

We thank the reviewer for this observation. Indeed, the actual speedup against the theoretical 2x hardware limit provides a much clearer picture of the solver's underlying computational capabilities. For this, we now oultine in the manuscript that Rosenbrock solvers involve extensive integer arithmetic for sparse matrix indexing, along with instruction fetching, and thus do not benefit from reduced floating-point precision, and cap the achievable acceleration below the theoretical limit.

The following text has been added in line 245:

“While the theoretical peak performance difference between FP32 and FP64 on the NVIDIA A100 is reported as 2× roughly (Table 1), the benchmarked speedup does not exceed 45% (Table 2). KPP-generated Rosenbrock solvers do not only perform floating-point arithmetic operations. Integration of chemical kinetics relies heavily on sparse matrix operations that require extensive integer arithmetic and array indexing. Additionally, adaptive time-stepping and other control-flow tasks within the solver require logical evaluations to be managed. These computations remain non-accelerated and require the same computational effort to be performed regardless of the employed precision. This is also reflected on memory traffic behaviour (Table 3). The substantial 63.24% reduction in L2-to-DRAM traffic indicates that FP32 is a more efficient fit for the GPU fast L2 cache. However, the same number of fetch instructions are needed to load/store the data. The GPU still spends time on other processes, none of which is sped up with the utilization of FP32, so the attainable kernel acceleration remains below the theoretical 2× hardware maximum.”

- Line 235 / Figure 7: It is unclear what generates the "distribution" of execution times shown for each node/GPU configuration in Figure 7. For instance, does each box/violin represent the set of individual per-timestep integration times (ts) accumulated over the course of a single 31-day production run (consistent with Eq. 3, AET = mean of ts over Nsteps)? The text introduces Eq. 3 immediately after this sentence but never explicitly links it to what is plotted in Fig. 7. Please state explicitly, either in the text or the Figure 7 caption, the source and sample size so that the spread shown in the figure can be correctly interpreted.

As indicated by the reviewer, explicit definition of the source of the execution time distribution is essential for correct interpretation of the violin plots. We have revised the text

to clarify that these distributions represent the individual per-timestep execution times accumulated over the entire production run.

The text in line 302 now reads as follows:

“The left panel (Fig. 9a) corresponds to simulations at T42 horizontal resolution, while the right panel (Fig. 9b) shows the results for the finer T106 resolution. Across all configurations and for both resolutions, the execution-time distributions for the FP32 implementation are consistently reduced compared to FP64, indicating markedly faster chemical kinetics integration when Single Precision is used. In terms of the average execution time (AET, defined in Eq. 6 as the mean of the per-timestep execution times t_s over the total number of steps N_{steps}), FP32 achieves a consistent ~ 30% improvement when compared to FP64, for all numbers of Nodes/GPUs used and for both model resolutions, demonstrating robust scalability of the single-precision implementation. This performance gain derives from increased throughput coming from the GPU architecture optimisation for single-precision floating-point operations.”

- Consider adding a test that fixes the hardware (number of nodes/GPUs) while varying the problem size — e.g., sweeping the box-model vector length parameter (VL_GLO), which the paper already uses for this purpose in Sec. 3.1 (fixed at 576,000). This would show whether the FP32 speedup reported in Table 2 is already saturated at the tested workload size or would continue to grow with more data per GPU, which is directly relevant given that the paper cites Kerkweg et al. (2025) finding higher speedups (5.14×–7.69×) for larger vectors of chemical species (L96-97). Establishing this relationship for the FP32-vs-FP64 comparison specifically would strengthen the scalability claims made in Sec. 3.2.1 and Fig. 8.

We thank the reviewer for this constructive suggestion and agree that workload saturation point is critical. To address this comment, we have added a new plot (Figure 3) that sweeps the vector length on a fixed hardware configuration (1 GPU) for all five Rosenbrock solvers. We have also added some discussion, clarifying that the performance flatlines beyond 300.000 grid cells, confirming that the reported FP32 speedup has reached its architecture-maximum potential.

The following discussion has been added at line 212:

“Figure 3 illustrates the solver throughput against varying vector lengths (VL_GLO) [5760, 5760e+2]. FP32 consistently achieves higher throughput against the double-precision implementation for all solvers and all size problems. For both precisions, throughput

increases rapidly with the number of grid boxes, indicating improved GPU utilisation. Our analysis demonstrates that single precision GPU chemical kinetics not only manages better time-to-solution when compared to double precision, but also scales more efficiently. FP32 in MEDINA is shown to achieve better utilisation of the GPU particularly for large problems relevant for production-scale atmospheric composition simulations.

Importantly, performance flatlines after 300,000 cells for both representations, indicating that GPU's capacity is saturated beyond this point and that the speed-up cannot be increased further, as the architectural limit of the hardware for this mechanism has been reached."

REFEREE 2 COMMENTS:

- The primary limitation I see is the restriction to NVIDIA devices. I would recommend including a brief discussion of why NVIDIA devices were chosen and any thoughts the authors might have about alternative device support and implications for the accelerated model's portability.

We agree that portability across different heterogeneous architectures is an emerging requirement for the ESM community. NVIDIA devices were targeted to leverage the existing MEDINA Legacy code and because of the availability and support on the JUWELS Booster supercomputer that was used for our simulations runs.

The following paragraph has been added to the manuscript (line 362):

"Medina v2.0 has exhibited great performance gains. However, the implementation currently only generates CUDA kernels and hence, only NVIDIA GPUs are supported. These devices were specifically targeted for this development due to the MEDINA legacy code that allowed for quick refactoring and development, and the availability of NVIDIA devices on the JUWELS booster supercomputer. As the HPC and Earth System Modelling community adapts different heterogenous architectures featuring AMD and Intel accelerators, portability is becoming a critical requirement. In the future, MEDINA, as an automated code-to-code compiler, can potentially address this limitation, by extending the framework to other programming models and open standards. This would ensure that the accelerated chemical kinetics simulations are vendor-agnostic and portable across different supercomputing environments and heterogenous architectures."

- Additionally, complementing the strong scaling analysis with a weak scaling analysis could support the authors' scalability assertions. Figure 8's two panels together act as a sort of informal weak scaling analysis, but a more formal weak scaling, e.g., co-varying resources with VL_GLO, would strengthen this argument.

We thank the reviewer for this suggestion, which strengthens our scalability assertions. We agree that a comprehensive weak scaling analysis will demonstrate whether the implementation suffers from communication overhead at scale and to address this, we conducted a weak scaling assessment with up to 64 GPUs (16 nodes), maintaining a constant workload per GPU while proportionally increasing the offloaded total grid size. Figure 4 was added to the manuscript, which demonstrates the processing rate and weak scaling efficiency.

The following discussion has been added in line 227:

“Figure 4 shows the weak scaling behaviour of the two representations, single- and double-precision, for the GPU implementation of the box model. The processing rate (P_N) increases with the number of GPUs, as shown in the left panel. The FP32 single precision implementation constantly sustains a higher processing rate compared to FP64 across all node configurations. Fig. 4 (right panel) presents the weak scaling efficiency (E_(weak)), with high scalability for both representations, as both manage an efficiency of over 99% for up to 64 GPUs. These results indicate that the execution time remains almost constant regardless of the total number of nodes employed, as the embarrassingly parallel nature of the problem means there is no communication overhead between GPUs.”

- Figs. 2, 6: Please include a measure of deviation from the 1:1 line, e.g. RMSE or NME. Visual inspection for deviations on a log scale spanning ~50 orders of magnitude obscures any actual deviations. Alongside RMSE or NME, something like an integrated difference over the box or domain could provide an idea of how much mass is over- or underpredicted compared to the baseline.

This is an insightful suggestion, and we agree that relying on log-log representation can obscure minor deviations across a vast range. We have introduced Normalized Mean Error (NME) and the integrated mass bias to address this comment. Both Figure 2 and Figure 7 have been updated to show the metrics, and we have expanded the manuscript text to discuss these statistical validations.

The following text has been added in Line 191:

“As these logarithmic scale scatter plots span 50 orders of magnitude, minor deviations might be obscured. Thus, two additional metrics are introduced to assess the solver’s fidelity: the Normalized Mean Error (NME) (Eq. 1) and the integrated bias (Eq. 2). The NME measures the absolute magnitude of the deviations relative to the reference, defined as:

$$NME[\%] = [\ \Sigma\ |x^{SP}_i - x^{DP}_i| / \Sigma\ x^{DP}_i\ ] \times 100$$

where x^{SP}_i and x^{DP}_i represent the final concentrations (or masses) of species i for the single-precision and double-precision implementations, respectively.

The bias is used to identify any systematic over- or under-prediction, which indicates an artificially created or destroyed mass. Bias is expressed as:

$$Bias[\%] = [\ \Sigma\ (x^{SP}_i - x^{DP}_i) / \Sigma\ x^{DP}_i\ ] \times 100$$

As displayed in the two plots, the NME is exceptionally low at 1.16 × 10^-5 %, which is also complemented by a negligible integrated bias of -3.7 × 10^-6 %, confirming that single precision manages to strictly conserve mass and does not systematically mispredict concentrations."

and in line 280:

"The visual agreement across the full domain is also confirmed mathematically with the statistical deviations NME (Eq. 1) and Bias (Eq. 2). As shown within the plot, these are exceptionally small, measuring at 1.19 × 10^-8 % for the former and at 1.12 × 10^-8 % for the latter, demonstrating that no systematic mass drift or numerical divergence occurs with the employment of single precision solver for global-scale simulations over extended periods."