Authors' Response to Referee Comment #1

on Manuscript "CLEO: The Fundamental Design for High Computational Performance of a New Superdroplet Model" by Clara J.A. Bayley et al. (2026)

Title	CLEO: The Fundamental Design for High Computational Performance of a New Superdroplet Model
Author(s)	Clara J.A. Bayley et al.
MS No.	egusphere-2025-4398
MS type	Model description paper

We thank Referee #1 very much for taking the time to review our manuscript and suggest improvements. We found their comments constructively critical as well as helpful for making our manuscript more accessible to a wider audience, most of all regarding our discussion of Monoids. We address the general comment in the following section and thereafter the minor comments and revisions.

Please note the updates to the dataset for this paper (https://doi.org/10.17617/3.LNRKSJ) are currently viewable from this link whilst they await publishing:
https://edmond.mpg.de/previewurl.xhtml?token=3eecde11-6c96-473f-9037-bfb58344d59b.

Response to the General Comment

The general comment made was the following: *This is an interesting manuscript, but very computer science oriented. It describes a framework for a new super droplet code. There is not much scientific content. Normally this would likely be normally this is supplementary material to an actual set of scientific simulations, even for Geoscientific Model development. It seems basically a proof of concept. It almost seems like this needs to be in a computer science journal. I am not sure how GMD should treat that. It's up to the editor on that front. There is nothing really wrong about the manuscript as a description of the technical aspects of a framework for a new superdoplet code. I also note that much of the computer science stuff is beyond my expertise. I was hoping for more scientific content. I'm not sure how ready for scientific content this code is.*

We understand this Referee is uncertain whether the content of this manuscript is relevant for GMD; nevertheless we feel strongly that it is. Indeed we intentionally wrote this paper to cover only the fundamental design (i.e. "the technical framework of our code"), and we kindly refer our readers to its companion paper: "Cleo: The Numerical Methods of a New Superdroplet Model including a Droplet Breakup Algorithm" by Bayley et al. (egusphere-2025-4399), for the scientific capabilities of our model. Our reasoning for dividing the scientific content and the computational into two separate manuscripts was exactly because the papers cover very different aspects of the model and therefore appeal to very distinct audiences. This paper appeals to the more computationally grounded audience, interested in the technical details and challenges of scientific programming for superdroplet models, whereas the companion to this paper is targeted towards the "scientist user" of the model, who is instead interested in the cloud microphysics which Cleo can model. Both papers are necessary to form a complete description of Cleo as a new superdroplet model. We feel our decision to describe Cleo's computational structure in such detail is well supported by the response of Referee # 2, which, in our opinion, demonstrates there is an audience who are greatly interested in the computer science underlying our model. We also agree with Referee #2 that the topics we cover in this manuscript are often missing in the cloud microphysics community's literature on model development, although they are of clear value and match the scope of GMD.

Response to Minor Comments

We thank the Referee for the minor questions and clarifications they posed. We have addressed all the "minor comments" individually in the manuscript, and provide additional explanation for what we changed in the following.

- *Page 1, L16: 'obscurity' is not the right word. Uncertainty?* We have deleted the word, the text now reads (P1 L17-20): "With the advent of Global Storm Resolving Models (GSRMs) which remove the need to parametrise convection, ~~obscurity in~~ cloud microphysics is now one of the leading sources of uncertainty in global models too (e.g. Miyakawa et al., 2014; Stevens et al., 2020; Suematsu et al., 2021; Bao andWindmiller, 2021; Lang et al., 2023; Takasuka et al., 2024; Naumann et al., 2025).".

Page 2, L41: However, SDM still requires use of collision / collection kernels, which are uncertain. . . .might need to note that. Bin schemes have the same issue (and bulk schemes don't represent this at all).

Yes we completely agree with this comment, and we discuss this extensively in the companion paper to this one: egusphere-2025-4399, e.g. allowing two different treatments of collisions (P5 L126-132), different kernels within these treatments (P6 L143-150), and also different outcomes (P7 L157-162). We have also amended the text in this manuscript to state (P2 L48-50): "Although this does not eliminate the epistemic uncertainty caused by insufficient knowledge of microphysical processes and condensate attributes, SDM's convergence property makes its physical interpretation both conceptually elegant and straightforward. This in turn helps us explore such knowledge gaps.".

- *Page 3, L65: Define CLEO with first use in text as well as the abstract.*

  In response to a comment by Referee #2, we've decided to remove the CLEO acronym entirely and rename "CLEO" to "Cleo". In that response we explain "[...] We think the acronym we gave Cleo was one of the key reasons the expectations [the manuscript set] did not align [with its content] and upon further consideration we also find this acronym not only unhelpful in this context but also restrictive for Cleo's future development (we may expand Cleo beyond cloud microphysics and/or use Cleo on non-exascale systems). We've therefore reverted to using Cleo as a proper noun, as it was originally intended to be, in tribute to the real-life "super-women", Cleopatra and the anonymous mathematican whose pseudonym was Cleo."

- *Page 3, L68: Define exascale computer*

  Certainly; we have added (P3 L75-76) "[...] exascale high-performance computers, computing at least 10^{18} floating-point operations per second.".

- *Page 3, L77: What other aspects of performance?*

  We now state them explicitly; we have added (P4 L94-96) "[...] whilst compromising on other aspects of aspects of performance, namely Single Instruction, Multiple Data (SIMD) parallel processing and scalability with increasing number of superdroplet attributes.".

- Here we respond to the two comments regarding monoids collectively. These were:

  - *Page 3, L82: Please describe what a monoid set is. Maybe with an example. See below.*
  - *Page 3, L82 and Page 9, L230: For the non-mathematically inclined. Can you give a simple example of a monoid? The description is not that clear. What are some binary operations? What is a semi group? What is an identity element?*

  We have re-written the first paragraph of the monoids section to address these comments. At the start of this explanation we rephrase the mathematical description of a monoid in simpler terms and thereafter we include the tangible example of acrylic paint. The paragraph now reads (P11 L291-297): "Cleo uses monoids to enable model flexibility without added run-time from conditional code branching. In abstract mathematics, a monoid is a closed set with an associative binary operation and an identity element. Put simply, monoid sets are composed of elements (members) which result in another element of the set when they're added pairwise together, and the identity element is defined such that the outcome of its addition with another element is just that other element unchanged. Acrylic paint is one example of a monoid set, whereby different colours are the elements of the set and transparent acrylic paint is the identity element. Mixing two acrylic paints together is their associative binary operation (in this special case, also commutative).". We have removed the word "semigroup" as we realised it is more confusing than helpful to non-expert mathematicians / set theorists.

- Here we respond to the two comments regarding advection collectively. These were:

  - *Page 7, L191: How will the array of super droplets with different grid positions be handled by advection? Is this a problem for efficiently if that part of the code wants to loop over grid boxes?*
  - *Page 7, L191 and Page 9 L227: Is the basic intent then to run the SDM on a different grid than the dynamics/? See comments above about advection. Can the SDM do its own advection if 1 way coupled?*

  We have added text clarifying how Cleo handles (superdroplet) advection in P8 L217-220 "Whereas Cleo advects superdroplets according to the fluid flow (wind fields) itself, to advect thermodynamic fields (temperature, pressure etc.) Cleo must be coupled to a host dynamical driver capable of advection. However the domain decomposition and computational resources for the dynamics can be independent from those used by SDM." as well as in P10 L275-279: "Cleo can enact microphysical processes on the superdroplets

in each grid-box and it can advect superdroplets according to Figure 3: first updating their coordinates and SD-GBX-Indexes and then re-ordering the superdroplet arrays. The numerical methods for microphysics and for updating superdroplet coordinates are provided in Bayley et al. (2025). Whilst Cleo can thus advect superdroplets, to advect thermodynamic fields as per fluid dynamics, Cleo must be coupled to a dynamics-solver with a fluid-dynamical core."

Specifically regarding the question *"Is this a problem for efficiently if that part of the code wants to loop over grid boxes?"*, we kindly refer the reader to the changes we made in response to the comment by Referee #2 regarding Kokkos Thread Parallelism.

Specifically regarding the question *Is the basic intent then to run the SDM on a different grid than the dynamics/?* Yes one part of the intent is that SDM can run on a different grid than the dynamics. However the other part, perhaps more important too, is that Cleo can run concurrently to the dynamics and hence Cleo can have a different domain decomposition. As stated in P10 L271-272: "Cleo is designed to run concurrently to a host dynamical driver called a "dynamics-solver" and exchange information with it via a "dynamics-coupler"" and hence (P11 L286-289) "In general, using MPI means Cleo and the dynamics-solver do not have shared memory and that their domain decompositions are independent. Whilst this can result in costlier communication, it also maximises our freedom to optimise the load balancing and so economise the allocation of computer resources.". The option for using a different grid as well as domain decomposition is in some-ways a bonus, which may allow the numerical methods for dynamics and SDM to be more accurate or easier to compute, as stated in P11 L284-286 "Using different grids is advantageous because it enables the dynamics-solver to compose the domain in the optimal way for its fluid-dynamics. Meanwhile Cleo can compose the domain more favourably for SDM, for example using a nested grid, or grid boundaries which reduce grid-box volumes and/or simplify the numerics of superdroplet motion.". (The drawback being that you need to interpolate between the grids during communication, but fortunately the YAC coupler does that well for us.)

- *Page 11, L289: But what if process A is estimated with a rate that would result in say complete removal of drops, and then halfway through that A step process B depletes more? How do you harmonize different process rates with different timeteps?*

  In P12 L312-317 we have added text to make it clearer that the monoids are associative but not necessarily commutative: "Note that since monoids are not necessarily commutative, "AB" is allowed to cause different microphysical outcomes to those of "BA" — in this example, the growth of superdroplets by condensation before collisions can result in different outcomes to collisions followed by condensation (i.e $AB \neq BA$). [...] [Cleo creates] one final object, Z, whose "do_microphysics" function is an assembly of all the microphysics we desire (again, associatively but not necessarily commutativity).". In your example, supposing at the current time-step both A and B are "on-step" i.e. should be called, then if process A is enacted first there will be the complete removal of drops such that when B is called it cannot remove any more. If process B where however enacted first, some droplets will be removed by process B and then A will remove the rest afterwards. To get the desired outcome of microphysics, be that A then B or B then A, one must choose to combine A and B in the correct order. This is irrespective of whether or not they have different time-steps, except that perhaps users may desire that processes with shorter time-steps are computed first (in your example, B then A not A then B).

- *Page 15, L319: How does Figure 5a relate to figure 5b? Not clear how figure 5b plugs into 5a/*

  We've added dashed arrows to connect Figure 5a and 5b together so the part where Figure 5b plugs into Figure 5a is now more easily identifiable.

- *Page 16, L360: This drop concentration is quite high and represents very polluted conditions. 100cm-3 would be more reasonable over land. Does that affect the results? You likely would get more precipitation. Are you trying to delay it?*

  We chose this aerosol number concentration because it is a reasonable value for marine aerosol conditions (Lohmann et al., 2016), although as you rightly point out it delays precipitation and would represent very polluted conditions over land. Delaying precipitation helps to keep the superdroplets in the domain longer and so have more time-steps of the simulation to average over when measuring the performance. The performance results, once normalised by simulated time-step as we have done, would be negligibly affected by lowering the droplet number concentration, we would just have less time-steps to average over.

- *Page 30, Figure 6: what time in the simulation is this? 80 min? Also, why does # superdroplets = # grid boxes? One per grid box?*

  Yes this is the cumulated trajectories of the random sample of superdroplets after 80 mins. We've clarified this in the figure caption which now reads: "An example of the performance tests with 16384 grid-boxes. (a-d) The initial conditions, (e) The growth and trajectories of a random sample of 500 superdroplets between

1500 m$< x <$ 3000 m up-to 80 mins for the test case with 4.194×10^6 superdroplets overall." as well as in the main text (P19 L459): "An example of the superdroplets' evolution up-to the end of the simulation is shown in Figure 6e.". Thank you for spotting the error in the figure caption regarding the number of superdroplets, we have corrected this and apologise for any confusion it may have caused.

Additional Minor Revisions

As well as taking on the feedback of the Referee, we have made some additional minor changes to the original manuscript listed here:

- P1, L19-20 and P2 L31-32: corrected references.
- P3 L89 and P20 L518: clarified meaning of the dependence of Cleo's MPI performance on the host-dynamical model by changing "the host dynamical model" to "the fluid-flow given by the host dynamical model".
- P12 L335-336: added an additional point regarding the benefits of templated code: "This allows compilers to efficiently inline and optimise the used code, whilst at the same time increasing code locality by not compiling unused code".
- P17 L410: "4 byte" rephrased "4-byte types".
- P21 L557: "minimum bound" rephrased "conservative estimate".
- Code listings: formatting adjusted to allow line-breaks.
- Code and data availability: citation formatting corrected.
- Acknowledgements: added another funding source and referees.
- References: Updated the companion manuscript citation (for egusphere-2025-4399).

References

Lohmann, U., Lüönd, F., and Mahrt, F.: An Introduction to Clouds: From the Microscale to Climate, Cambridge University Press, ISBN 9781139087513, https://doi.org/10.1017/CBO9781139087513, 2016.

Authors' Response to Referee Comment #2

on Manuscript "CLEO: The Fundamental Design for High Computational Performance of a New Superdroplet Model", now renamed "Cleo: The Fundamental Design of a New Superdroplet Model for High Computational Performance (v0.39.0)" by Clara J.A. Bayley et al. (2026)

Original Title	CLEO: The Fundamental Design for High Computational Performance of a New Superdroplet Model
New Title	Cleo: The Fundamental Design of a New Superdroplet Model for High Computational Performance (v0.39.0)
Author(s)	Clara J.A. Bayley et al.
MS No.	egusphere-2025-4398
MS type	Model description paper

We thank Referee #2 very much for taking the time to review our manuscript and suggest improvements, we are pleased with the level of detail they have gone into and we found their feedback both supportive and constructive for the manuscript. In the next section we address all of the general comments which the referee made; in the section thereafter we address the "other" comments.

Please note the updates to the dataset for this paper (https://doi.org/10.17617/3.LNRKSJ) are currently viewable from this link whilst they await publishing:
https://edmond.mpg.de/previewurl.xhtml?token=3eecde11-6c96-473f-9037-bfb58344d59b.

Response to General Comments

We thank the Referee for the general comments they made on the manuscript. Here we address each of them individually and to aid readability we have re-arranged them in chronological order.

Title

- *The title is missing software version, which is required by GMD guidelines. Moreover, the title suggests that a new model is introduced, but the paper outlines a new implementation of a "classic" model.*

  We have renamed the title to include the version of Cleo presented in this paper, it is now "Cleo: The Fundamental Design of a New Superdroplet Model for High Computational Performance **(v0.39.0)**". In the second point below regarding the introduction, we also explain our decision to remove the acronym for "CLEO" and name the model simply as "Cleo". In the title we have chosen to describe Cleo as a "superdroplet model" rather than a "superdroplet method model" to make sure it is readable, however, in the main text we make clear Cleo is a is a new model which uses the (not-new) superdroplet method, e.g. (P3 L79): "Hence we are creating Cleo: a novel implementation of SDM [...]", and we also now more precisely define "SDM" as the superdroplet method (see first comment under "General" below, regarding the SDM acronym).

Abstract and Introduction

- Here we address the two points regarding the abstract and introduction together, these were:
  - *Starting with the abstract, there are several bold statements regarding optimal performance, efficient access patterns, cache efficiency, load balancing, and speed-ups, which later on in the text are clarified as based on intuition rather than comparison against alternative implementations. Matching the abstract and introduction with the content of the paper will improve reception.*
  - *The abstract focuses on "marketing", is repetitive, and does not convey a summary of what is in the paper (e.g., what kind of simulations were performed); suggest rewriting focusing on the key results from the presented development, and avoiding vague phrasing (e.g., a diverse range of computer architectures, a high degree of flexibility, ready to be used for understanding warm-cloud processes; all these would best be changed from "a"*

to "the" statements: which architectures, what kind of flexibility, which processes and what do you mean by understanding?)

We thank the referee for pointing out the Abstract did not well convey the content of the paper and the wording was often too vague and/or bold. We've completely re-written the abstract in order to take on board these criticisms, and we think the Abstract now more correctly summarises the paper and its wording is much more precise. Also to better match with the content of the paper, we've softened the wording in P3 L94- P4 L99 of the introduction: "we chose Cleo's memory layout **with the aim of** economically allocating memory and optimising memory access patterns for loops over superdroplets, whilst compromising on other aspects of performance, [...]. Secondly, we have **sought to** maximise Cleo's freedom to allocate resources on massively parallel heterogeneous computer architectures; [...], **in doing so we aim to be able to allocate resources efficiently**."

- *Furthermore, I suggest to make it clear upfront in the abstract and introduction to the paper that the development - in the stage presented in the paper - is not tested at exascale, not even in a multi-node setup, and only with one-way coupling with a prescribed-flow driver*

  We thank the referee for encouraging us to better align the expectations set by the introduction and abstract with the content of the manuscript. We think the acronym we gave Cleo was one of the key reasons the expectations did not align and upon further consideration we also find this acronym not only unhelpful in this context but also restrictive for Cleo's future development (we may expand Cleo beyond cloud microphysics and/or use Cleo on non-exascale systems). We've therefore reverted to using Cleo as a proper noun, as it was originally intended to be, in tribute to the real-life "super-women", Cleopatra and the anonymous mathematician whose pseudonym was Cleo. The first paragraph introducing Cleo now reads (P3 L79-81) "Hence we are creating Cleo: a novel implementation of SDM aiming to model warm-cloud microphysics efficiently on top-tier and exascale high-performance computers. Cleo's name is a tribute to two "super-women", Cleopatra VII Thea Philopator and the pseudonymous mathematician Cleo."

  Additionally, we have made several amendments to the paragraph thereafter (P3 L82- P4 L99) to make it very explicit what this paper does and does not contain, most notably the second sentence of the second paragraph states "Specifically, this paper explains how we exploit the massively parallel heterogeneous resources and the hierarchical memory layout of individual nodes of high-performance computers.", and P3 L89-92 "In a follow-up paper we will therefore present Cleo's distributed-memory parallelism (MPI domain decomposition) and evaluate its performance specifically for LES **across multiple nodes of** top-tier and exascale high-performance computers. With regard to Cleo's fundamental computational structure **on single nodes**, [...]". We've also amended P20 L518-521 along similar lines to state: "As such, we will evaluate Cleo's distributed-memory parallelism in an follow-up paper specific to LES conducted across multiple nodes of top-tier and exascale high-performance computers. This paper meanwhile discusses Cleo's fundamental computational design and it's performance on shared-memory architectures (single nodes)."

Edmond dataset(s)

- *the Edmond dataset metadata keywords misleadingly include: Climate Modelling, LES, Exascale Computing, ICON;*

  We thank the reviewer for pointing this out and we agree it is indeed misleading. We originally chose to include those keywords because our current work and future publications is/will be relevant to those keywords; however we realise that does not befit this dataset and this manuscript's fundamental model description. We've therefore replaced the keywords with "Superdroplet Model, SDM, Cloud Microphysics Modelling, HPC" and "Cleo".

Kokkos Thread Parallelism

- *mention alternative parallelisation strategies for the SDM (e.g., where Monte-Carlo collisional growth parallelisation is done in parallel across all pairs in multiple cells of a given subdomain, rather than - IIUC - serially within cell);*

  The parallelisation is done over superdroplets, over grid-boxes, and over superdroplets in grid-boxes hierarchically whenever desired. The strategy for parallelism is flexibly determined by Kokkos depending on the architecture, or manually by the user. We've now made this clearer with additional text in P9 L230-234: "When loops over superdroplets are nested inside loops over grid-boxes, as occurs during microphysics and superdroplet motion but not necessarily during initialisation or data output, hierarchical parallelism is invoked;

that is both the outer loop over grid boxes and the inner loop over superdroplets are parallelised. The exception is if Kokkos (or manually the user) determines that for the given hardware available the number of grid-boxes or superdroplets is too small for it to be worthwhile to parallelise the loop."

MPI Domain Decomposition

- *elaborate on the technological challenges stemming from the spatially uneven resource requirements of particle-based simulation (i.e., grid cells with clouds vs. grid cells without clouds), and from the fact that the employed simulations inherently cover aerosol budget (implementing aerosol sources should also benefit from the proposed memory layout);*

  Based on this feedback, we've greatly elaborated on our discussion of MPI domain decomposition in Section 3.2 (P9 L239- P10 L269), especially in lines P10 L248-262 where we discuss specifically the technological challenges of particle-based, in our case superdroplet, simulations with regard to load balancing across computational resources. In P10 L254-58 we include specific mention that the choice of boundary conditions, including aerosol sources as well as lateral sinks of superdroplets, are one source of challenge because they can unbalance workloads: "The major challenge with such superdroplet-based domain decompositions is that superdroplets (and cloudy regions) move around the domain and superdroplets may be added or removed depending on a simulation's boundary conditions (e.g. its aerosol/superdroplet sources and lateral superdroplet sinks). The consequent changes in the number and activity of superdroplets changes the workload of MPI processes and a better optimised domain decomposition would dynamically adapt to restore balance.".

Monoids

- *mention alternative technologies as how they compare to the employed C++20;*

  We presume this point mainly refers to the section describing our computational monoids since the other infrastructures/software we mention (arrays, classes/structures, MPI, YAC and Kokkos) are not C++ specific. We've now re-structured the section which introduces monoids (see next bullet point) and included a new paragraph (P12 L327-333) discussing what other technologies would also be suitable as opposed to C++20. This paragraph reads: "There are many other plausible ways to express monoids in code. Within older C++ standards (pre-2020) one could instead use for example a base class, a trait, or even just constraints written on a piece of paper. However, base classes would create more restricted, closed definitions of monoids, and the latter options would be more verbose and error-prone. Based on how we defined computational monoids above, other compiled or just-in-time compiled languages could also implement monoids computationally with the same overall behaviour as our monoids in C++. Naturally, languages with object-oriented programming paradigms are better-suited to implementing monoids since monoids are fundamentally described by set theory.".

- *Given that the paper's intended audience seems to be tech-aware research software engineering community, I suggest to redact the technical parts of the paper with references to alternative technologies. For instance, the "flexibility" featured in the title of Section 4, would likely be termed "workarounding the stiffness of C++" from a just-in-time compiled language perspective. Covering it in the paper is fine, but I suggest rephrasing - making sure that the paper is readable, approachable and appreciable for audiences ranging from (modern) FORTRAN to Julia coders. For instance, when writing about C++20 concepts, it would be apt to refer to Julia multiple dispatch alternatives.*

  Also here we presume that "technical parts" is primarily referencing to our section on monoids because we are reasonably confident that the other technical parts (arrays, classes/structures, MPI, YAC and Kokkos) are already familiar with our "tech-aware research software engineering community", particularly climate model and LES developers, and that they are understandable to (modern) FORTRAN and Julia coders even if they do not know the details of YAC or Kokkos. If this is not the case, we are happy to be pointed to any other specific technical parts of the manuscript which are not yet satisfactory so that we can improve them.

  To address this referee's point in regard to monoids, we've restructured the section introducing monoids and we have generalised the first mention of computational monoids therein to be agnostic of C++ so that it appeals to a wider audience (P11 L297-304): "We apply the idea of monoids computationally by creating objects in code that can be combined pairwise and associatively with one-another. To do this, first we define a set of constraints a code-object must satisfy in order to be allowed to be combined with another such object. In other words, we define constraints an object must satisfy to be part of a certain code-based monoid set. Secondly, alongside these constraints, we define a function (an associative binary operation) which determines exactly how two objects which obey these constraints are combined together such that the return of the function is a new object which itself satisfies the given constraints. In this way, any number

and permutation of such objects can be combined sequentially to create the final object that is used at run-time." . In combination with the rest of the section, we think this also makes clear that the section is truly about flexibility through monoids, in any language but in our case C++, and not about working around C++ stiffness. Later on we then describe our specific C++ implementation with template-metaprogramming and C++20 concepts (P12 L321-326): "To implement monoids in Cleo, we take advantage of template meta-programming with C++20 concepts. Specifically, for a given monoid set we use a C++20 concept to impose the constraints of the set on templated types, and we overload the right-shift operator (`operator>>`) to implement the function for the associative binary operation. We then create structures/-classes which obey the C++ concept and can thus be combined during compilation, in whatever way desired, to instantiate the templated type that is used throughout the code at run-time. Using C++20 concepts is not essential, but has the added benefit of more comprehensible error messages than ordinary template meta-programming in C++." and following this we include the new paragraph on alternative technologies as mentioned in the bullet point above.

We have purposefully not generalised the subsequent sub-sections (Section 4.1 and Section 4.2) which describe in detail the monoids Cleo defines for microphysical processes and observers. This is because those monoid definitions are specific to Cleo and the purpose of those sections is not to define general monoids for microphysics and data output. The purpose of those sections is rather to explain the specific implementations of the monoids that are essential for understanding how to use and develop Cleo, as is of course necessary to adequately describe Cleo in its model description paper. We have modified P13 L344-L348 to be more clear that the following sections are Cleo specific: "Thus, to ensure run-time performance whilst still enabling maintainability and flexibility, Cleo defines monoids for microphysics and data output. Section 4.1 details our monoid for microphysical processes, namely the C++ concept which defines microphysical processes, the structure which acts as the identity element of the set, and the right-shift operator used to combine microphysical processes with one another. Likewise, Section 4.2 details the monoid set for "observers", the elements of which can be used for data output."

- *elaborate on the implications of the implementation choices in terms of maintenance (e.g., the Monoids should facilitate unit testing, although the contents of the 'tests' folder in CLEO does not seem to leverage it yet...).*

  We added on P12 L339-343: "[...] branching would add computational expense and reduce the code's modularity. Modular as opposed to monolithic code is more readable and easier to maintain because it can be understood in piecewise units and each unit can be tested and developed independently, as well as concurrently. Although at the time of writing Cleo does not yet have unit tests for its monoids, they are still modular units of code and thus facilitate code readability and maintainability." . You're quite right to point out that we haven't yet found the time to implement unit tests in Cleo, we currently use the examples as integrated test cases instead (see the examples folder and our documentation for details: https://yoctoyotta1024.github.io/CLEO/usage/examples/examples.html). We realise this is by no means ideal, nor best practise, and we do intend to write unit tests in the near future. Since we don't yet have them, it would be inappropriate to be specific about our monoids facilitating unit testing in our code yet; nevertheless they do facilitate making Cleo maintainable and readable in the sense they make Cleo modular and hence easy to read and edit/develop.

General

- *The SDM acronym was introduced by Shima et al. as Super-Droplet Method; here it is deciphered as Super-droplet Model (not Method) - is it intentional? Also, it seems unclear what the authors mean by SDM. On page 1, lines 31-32, SDM is used in a broad sense of super-particle microphysics with works employing contrasting particle-collision representations cited (probabilistic, deterministic, super-particle-conserving and not), and on page 17, lines 418, SDM is used to label three works not modelling particle collisions at all; on page 1, line 42, SDM is associated with "its coalescence algorithm" implying that SDM is used in the specific sense of Shima et al. 2009 Monte-Carlo algorithm. Yet, on page 6, line 146, there is a statement about "CLEO's SDM algorithms" (plural). This is in many ways misleading. Please state at the beginning, what is meant by SDM: particle-based microphysics methods (which predate SDM, and would likely better be called particle-based microphysics), Shima et al. Monte-Carlo algorithm; the whole set of schemes from the SDM paper, or something else? This is also important given the later conclusions about random shuffling bottlenecks - the paper does not provide enough context to understand where, when and why the shuffling is needed.*

  Thank you for pointing this out, we had not thought carefully enough about how we define SDM. We've now corrected the first mention of the acronym (P2 L36) to be "The Super-Droplet Method (SDM; Shima et al., 2009), [...]" and clarified how we use the term "SDM" (P2 L42-48): "Several SDM implementations already exist, (e.g. Shima et al., 2009; Arabas et al., 2015; Naumann and Seifert, 2015; Brdar and Seifert,

2018; Jaruga and Pawlowska, 2018; Shima et al., 2020; Chandrakar et al., 2021; Bartman et al., 2022; de Jong et al., 2023; Matsushima et al., 2023) and are similar to other particle-in-cell-based cloud microphysics models (e.g. Sölch and Kärcher, 2010; Andrejczuk et al., 2010; Riechelmann et al., 2012; Hoffmann et al., 2015), the key difference being SDM's profound convergence property. As the number of superdroplets increases, the model, including its algorithm for droplet collision-coalescence, tends towards what we expect from explicit individual particle simulations (Shima et al., 2009). ". We thus clarify that by SDM we mean the Lagrangian particle-in-cell-based microphysics with the collision-coalescence algorithm as per Shima et al. 2009. Correspondingly we've updated P20 L509-510 to point to papers which use superdroplets according to that paper "[..] for example in studying the role of turbulence during droplet growth (Li et al., 2018, 2020; Grabowski and Thomas, 2021; Chandrakar et al., 2024; La et al., 2025)." (the incorrect citations where human-error citing papers of the same first author but incorrect year, thank you for spotting this!).

We've also clarified in P6 L164-165: "the algorithms Cleo uses to conduct SDM microphysics" and P19 L463-464 we've clarified that the shuffling is required in order to enact SDM collision-coalescence: "This is because the bottleneck of microphysics is the random shuffling of superdroplets required by the method for SDM collisions".

- *Figures 1, 2, 3, 6, 7, 8 and 9 are supplied in raster graphics format, please replace all figures with vector graphics suitable for publication. Replotting the figures, ensure that font size roughly matches the font size in the main body of the paper text - currently most labels in the plots are too small.*

  We have re-made all the figures as .pdf files, and increased all the font-sizes.

Response to Other Comments

We thank the Referee for the other comments they made on the manuscript. We've addressed each of these individually with the changes shown in the tracked changes document. For some comments, we provide additional responses in the following:

- *abstract: "conservative memory usage" is mentioned; however, first: it is not explained in the paper how an alternative memory usage would imply more memory needs (in contrast, it is mentioned that the employed sorting algorithm doubles the memory footprint!); second: the paper states that CLEO abstracts away such details as if the storage layer is based on linked lists or contiguous arrays.*

  This comment no longer applies to our re-written abstract. Nevertheless, we would like to clarify here why we used the term "conservative memory usage" to describe Cleo originally. We used the term because (P6 L166-169; now it's own paragraph to make more clear) "Single, contiguous arrays of grid-boxes and superdroplets efficiently handle memory allocations. As opposed to having superdroplets scattered in memory, e.g. by having a separate array of superdroplets for each grid-box, having a single array of superdroplets requires less frequent memory (de)allocations when superdroplets move around the domain and therefore also avoids the need for buffers which would increase the memory consumption.". Having said that, it's unclear if this saves memory overall because, as rightly pointed out, the current sorting algorithm we use doubles the superdroplet memory footprint.

  The second part of this comment regarding Cleo's storage layer also no longer applies to the rewritten abstract. Nevertheless, to resolve any confusion the previous abstract may have caused we would like to respond here. Cleo uses contiguous arrays (specifically Kokkos::View and Kokkos::DualView instantiations, see https://kokkos.org/kokkos-core-wiki/API/core/view/view.html). Their layout (right, left, stride, or even custom defined) is indeed abstracted, but the allocation is not. We state on P5 L157-158 "the structures for each grid-box are stored together in a single array (an Array of Structures; AoS), and likewise superdroplets are stored in their own AoS, in a contiguous block of memory completely separate from grid-boxes." and then on P9 L225-228 we state "[Kokkos determines] how to appropriately allocate the memory and execution resources to achieve high performance. For example, it determines whether to use column- or row-major layouts for arrays of grid-boxes and superdroplets in order to avoid cache misses, and it chooses task sizes for threads of parallelised loops which optimise cache blocking.". We mention linked lists only in the discussion of how grid-boxes "view" the superdroplets associated with them at any given time as an alternative to our current method using pointers (strictly speaking a Kokkos::Subview): (P5 L132-134) "[The view] is an indicator to their location(s) elsewhere in memory. For example, the view could be a linked-list of pointers to individual superdroplets, or it could be two pointers to the start and end of a sub-section of a larger array of superdroplets (as it is in our current implementation).".

- There were several comments regarding the first paragraph of the manuscript on P1. These were:

- *page 1 / line 13: "observations of warm-rain" is a very broad statement; this very first sentence of the paper would benefit from clarification on what kind of observations you have in mind, what is the type of disagreement (total amount, timing, size spectrum?). For instance, the abstract of the vanZanten work cited to support the disagreement statement states: "The ensemble average of the simulations plausibly reproduces many features of the observed clouds..."*
- *page 1 / line 16: "obscurity in cloud microphysics" calls for elaboration; obscurity suggests a lack of process understanding (next sentence also mentions "fundamental gaps in our microphysics knowledge"). It is fair to say we don't know everything about cloud microphysical processes, but especially given that the study deals with warm-rain clouds, it would be best to explain what is unknown and how much are the knowledge gaps limiting us vs. how complex are the challenges in modelling well understood but multi-scale and non-linear processes.*
- *page 1 / first paragraph: overall, I find the first three sentences (with 15 references) "obscuring" the idea of the paper, rather than introducing it. Clearly subjective, but perhaps the two above points can help in rephrasing it.*

We've clarified we mean macrophysical observations of warm-rain, i.e. its spatial and temporal pattern (P1 L15-17) "Persistent and large discrepancies between observations of the spatial and temporal patterns of warm-rain and Large Eddy Simulations (LESs) have been ascribed to cloud microphysics for decades (e.g. Randall et al., 2003; Abel and Shipway, 2007; Ackerman et al., 2009; vanZanten et al., 2011) (see also summaries in Khain et al., 2015; Morrison et al., 2020)." . In the sentence thereafter we've removed the words "obscurity in" so the sentence states (P1 L18-19): "With the advent of Global Storm Resolving Models (GSRMs) which remove the need to parametrise convection, ~~obscurity in~~ cloud microphysics is now one of the leading sources of uncertainty in global models too" .

The aim of this paragraph was to emphasise there are two distinct sources of uncertainty in conventional, Eulerian cloud microphysics schemes and these stem from knowledge and model uncertainty, but we realise we hadn't managed to make this point properly. To make it better we've added the sentences (P1 L20- P2 L25) "Much of the uncertainty in both LES and GSRMs arises from epistemic uncertainty, that is uncertainty caused by our insufficient knowledge of microphysical processes and condensate attributes. However, distinct from that, these models also suffer from uncertainty caused by the Eulerian microphysics schemes we conventionally use to represent the knowledge we already have (Morrison et al., 2020)." This also points out these are Eulerian models to contrast with the Lagrangian superdroplet method later on.

- *page 1 / line 21: super-particle model is later on labelled as Lagrangian, perhaps labeling the "conventional" methods as Eulerian could help in pointing out the difference?*

  Thank you for the suggestion, we have included it in the sentence already mentioned above (P1 L23-24) "However, distinct from that, these models also suffer from uncertainty caused by the **Eulerian** microphysics schemes we conventionally use [...]" and in P2 L26: "Conventional, **Eulerian** bulk/moment and bin microphysics schemes have intrinsic deficiencies, [...]" .

- *page 1 / line 43: the commonly used term is "embarrassingly" rather than "extremely parallelisable"; given the scope of the paper, it seems reasonable to point here out that the reason for it is the non-overlapping pair sampling, while the GPUs enable one to leverage it.*

  We've corrected "extremely" to "embarrassingly" and added more explanation on P2 L52-54: "Moreover SDM is embarrassingly parallelisable because its algorithms for microphysics and motion act independently on individual superdroplets or non-overlapping pairs of superdroplets, [...]"

- *page 1 / line 46: this statement applies to bin models only, while "conventional" has been associated with both bulk and bin earlier; moreover it seems worth mentioning that for "bin" models, two attributes are already a rarity (Lebo & Seinfeld 2011), while employing more attributes is needed to resolve mixed-phase processes or chemical ageing in the context of aerosol-cloud interactions.*

  We've amended "conventional" to "bin" in the statement (P2 L56-60): "Its scaling with increasing microphysical complexity is also better than that of **bin** models once the number of superdroplet attributes is greater than about four (Shima et al., 2009)." , and also added that "This makes SDM better suited to modelling droplets with more complexity, which is rarely done in bin and bulk schemes and is necessary to better represent important microphysical processes, for example riming in mixed-phase clouds (Brdar and Seifert, 2018) and aerosol chemistry in aerosol-cloud interactions (Jaruga and Pawlowska, 2018)."

- *page 4 / line 118: suggest splitting SCALE and ICON references into separate parentheses;*

  Done (P5 L136-137), the text now reads "This is because LES grids can vary substantially, for example SCALE uses a Cartesian Arakawa-C grid (Sato et al., 2015; Nishizawa et al., 2015), whereas ICON uses a icosahedral-triangular Arakawa-C grid (Hohenegger et al., 2023), [...]" .

- *page 6 / line 153: "sequentially" suggests serial execution*

  This is certainly not what we meant (quite the opposite!) and upon reviewing this sentence we realise it lacked a proper explanation of the benefits to memory access patterns of our memory layout. We've therefore added considerable explanation on P7 L170-185, stating "One advantage is that we maximise locality of reference, [...] The second major advantage [is that our memory layout causes] less contention in memory access, [...] In many respects our memory layout amounts to cache blocking, [...]". This re-written paragraph also removes the misleading use of the word "sequentially".

- *page 7 / line 170: I do not agree with the statement "SoA layout would need to perform sorting/shuffling on the array for each individual sub-component of the superdroplets separately", since an SoA layout could feature an indirection layer mapping particle id through a (single) permutation vector;*

  We've clarified we meant "any SoA layout which still maintains the advantages for memory access patterns of ordered superdroplet data" (P8 L196-199).

- *page 7 / line 189: "to advect thermodynamics" sounds too jargonic (and suggests that it is CLEO which does the advection, but she is not IIUC);*

  We've clarified what we mean on P8 217-218 "Whereas Cleo advects superdroplets according to the fluid flow (wind fields) itself, to advect thermodynamic fields (temperature, pressure etc.) Cleo must be coupled to a host dynamical driver capable of advection" as well as in P11 L278-279 "Whilst Cleo can thus advect superdroplets, to advect thermodynamic fields as per the fluid flow, Cleo must be coupled to a dynamics-solver with a fluid-dynamical core.".

- *page 16 / line 363: please elaborate on how a "2-D kinematic flow" was adapted to a 3-D domain?*

  We've clarified on P18 L452-453: "To extend the simulation into 3-D, we replicate the 2-D (x-z) flow uniformly along the additional (y) dimension. "

- *page 16 / line 364: feedback from microphysics is a different thing than relaxation - likely "rather than" is a wrong phrase here;*

  "rather than" has been corrected with "nor" (P18 L454): " [...] without feedback from microphysics **nor relaxation** towards the initial profiles.".

- *page 16 / line 381: suggest rephrasing "suffers from microphysics" and "suffers from motion";*

  we rephrased "suffers from" to "is/are limited by" (P19 L471-472): "CUDA speed-up is limited by microphysics (Figure 8b), while OpenMP and C++Threads speed-ups are limited by motion (Figure 8c).".

- Here we respond to the two comments regarding serial vs parallel random shuffling and random number generators together. The comments were:

  - *page 17 / line 374: if only serial random shuffling is featured, it is worth highlighting earlier on;*
  - *since serial shuffling of particles has been identified as a bottleneck, parallel alternatives for permutations (e.g., MergeShuffle, arXiv:1508.03167) or parallel pseudorandom number generation (e.g., cuRAND for CUDA) should be considered... but on page 17, line 406 "creation of thread-safe random number generators" is mentioned - unclear;*

  Shuffling is inherently expensive, even when parallelised, and we mention this earlier in the manuscript on P3 L91-93: "With regard to Cleo's fundamental computational structure on single nodes, the memory and particle transport expenses are prevalent when accessing, sorting, and shuffling superdroplets." and in the discussion of Cleo's memory layout we state P8 L196-197: "the speed of algorithms which sort/shuffle superdroplets during motion/microphysics scale only with the number of superdroplets.". With regard to considering parallel alternatives for permutations, we have now stated more clearly that a different algorithm may improve performance on P19 L474-477: "The bottleneck is again the shuffling algorithm we use during collision-coalescence, presumably due to the relatively slow clock-speed of a single GPU thread during the serial step of the Fisher-Yates algorithm we use for shuffling. A different (possibly parallelised) algorithm may therefore improve performance," and P21 L549-550 "Using a different shuffling algorithm may be more optimal for GPUs, [...]", but we have now removed the first mentioning of the shuffling algorithm being serial in P19 L464 "[...] the bottleneck of microphysics is the ~~**(serial)**~~ random shuffling of superdroplets [...]" and we have softened the wording around suggesting it be parallelised ("(possibly parallelised)") because we feel the emphasis on a parallel as opposed to serial shuffling algorithm here is misleading.

  Considering this further, the MergeShuffle parallel random shuffling algorithm does not offer any significant speed-up over the Fisher-Yates serial shuffling algorithm for problem sizes less than 10^7 (arXiv:1508.03167

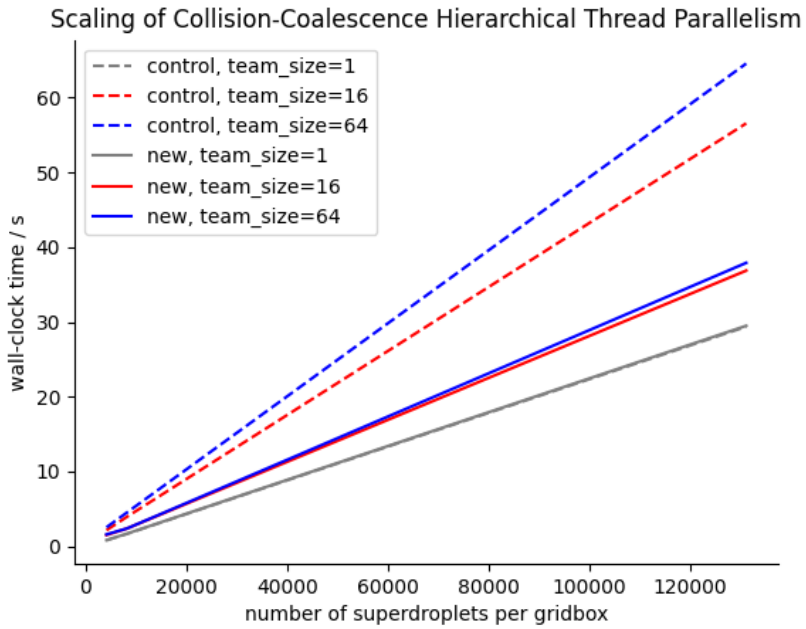

Figure 1: time for shuffling in a box-model performance test of the serial Fisher-Yates algorithm ("control"; dashed) and MergeShuffle ("new"; solid) for different numbers of CPU threads ("team_size" = 1, 16 or 64). In the case of Fisher-Yates with multiple threads, a single thread performs the shuffling and then synchronises with the other threads afterwards. The single-thread, i.e. serial, implementations of Fisher-Yates and MergeShuffle show the best performance here.

Fig. 1). We shuffle the superdroplets for each grid-box separately to enact collision-coalescence in each, distinct, grid-box (i.e. volume) and it would be *extremely* extraordinary to have 10^7 superdroplets per grid-box — we usually have no more than $\mathcal{O}(100)$. We therefore wouldn't expect to see any improvement by using this parallel random shuffling algorithm. Nevertheless, we tested the MergeShuffle algorithm in a box model but indeed we found the parallelised MergeShuffle algorithm does not offer any significant speed-up over the standard Fisher-Yates serial algorithm (Figure 1); moreover we found that with multiple threads, albeit CPU ones, it even performs slower than the single-thread tests. We choose not to mention this in the manuscript, mainly because it would distract from the focus of analysing Cleo's performance, and is not necessary for the conclusions of the performance analysis, most importantly that whilst Cleo's shuffling algorithm is the bottleneck of GPU builds on a single node, it is unlikely to be the bottleneck of larger, more complex simulations and so before conducting any further optimisation of the shuffling we should first assess Cleo's performance with the transport of superdroplets between nodes, additional microphysics, and/or two-way communication with a dynamics-solver with a fluid-dynamical core.

With regard to considering parallel pseudorandom number generation: our thread-safe random number generators do indeed generate psuedorandom numbers in parallel and we have added the word "parallel" in their description to clarify this on P20 L497 and P20 L499: "thread-safe **parallel** random number". (Note as per the Kokkos documentation: "in contrast to CuRAND, none of the functions of the pool (or the generator) are collectives, i.e. all functions can be called inside conditionals."; https://kokkos.org/kokkos-core-wiki/API/algorithms/Random-Number.html).

- *page 17 / line 411: "can divided" missing "be";* Corrected on P20 L502.
- *page 19 / lines 472-475: the "Code availability" and "Code and data availability" sections should be merged, and the licensing terms of the code and its key dependencies should be stated.*

  We have merged the two sections and re-written them following the guidelines from the GMD data policy (https://www.geoscientific-model-development.net/policies/code_and_data_policy.html). The core dependencies of Cleo v0.39.0 are: a GCC or Intel compiler for C++20, CMake, Kokkos, and yaml-cpp, which is also stated in Cleo's documentation and now in the dataset too. The full code and data availability section now reads "The current version of Cleo is available from its GitHub page: https://github.com/yoctoyotta1024/CLEO under BSD 3-Clause license, alongside its documentation: https://yoctoyotta1024.github.io/CLEO. Version v0.39.0 is described and tested in this paper and all the code, including this Cleo version, as well as the results

included this paper are archived in the dataset on Edmond under DOI https://doi.org/10.17617/3.LNRKSJ (Bayley, 2025). The core dependencies of Cleo v0.39.0 are: a GCC or Intel compiler for C++20, CMake, Kokkos, and yaml-cpp."

- In response to the comments regarding the formatting and references which were:
  - *in references, 19 entries have malformed URLs: https://doi.org/https://doi.org/...*
  - *in references, there are discussion-stage papers cited, for which accepted peer-reviewed papers are available: Yin et al. 2023, Matsushima et al. 2023;*
  - *in references: please double check bibliography formatting, e.g., the Takasuka et al. 2024 paper has its 2023MS003701 id given four times*

  We have fixed the formatting of the references and updated the entries for discussion-stage papers that are now accepted peer-reviewed papers.

Additional Revisions

As well as taking on the feedback of the Referee, we have made some additional minor changes to the original manuscript listed here:

- We have re-phrased the cost estimate in P21 L556- P22 L564 to compare with ICON's strong-scaling and to assume that cost overhead could also come from communication with the dynamics-solver, not just particle transport via MPI. We also consider the "extremely conservative estimate" to be a factor of 100 slower rather than 1000. The cost estimate now reads: "For a conservative estimate on the cost, let us assume that the overhead from particle transport via MPI and communication with the dynamics-solver increases Cleo's wall-clock by a factor of 10. Then, by using 1000 NVIDIA A100 GPUs (250 "Levante-like" GPU-nodes), the SDM setup from the performance tests presented here suggest that $\mathcal{O}(10^7)$ superdroplets per MPI process would need approximately one second of wall-clock time for every second of simulated time and have an order of magnitude less memory consumption than the maximum on each node. That would make Cleo's throughput approximately a factor of 10 less than the strong-scaling limit of ICON (Klocke et al., 2025), and even for an extremely conservative estimate, where Cleo's performance is slower by a factor of 100, a simulation of several hours that takes several weeks to run is still conceivable."
- Added more references for largest superdroplet simulations known to this author on P3 L73. Now: "(Sato et al., 2017; Chandrakar et al., 2021; Matsushima et al., 2023; Yin et al., 2024)" instead of "(Sato et al., 2018; Shima et al., 2020; Matsushima et al., 2023)".
- P20 L525 added "try to minimise cache loading and contention".