An efficient Bayesian approach to learning droplet collision kernels: Proof of concept using "Cloudy", a new n-moment bulk microphysics scheme

Melanie Bieli[1], Oliver Dunbar[2], Emily K De Jong[1], Anna Jaruga[1], Tapio Schneider[2], and Tobias Bischoff[1]

[1]Caltech
[2]California Institute of Technology

November 26, 2022

Abstract

The small-scale microphysical processes governing the formation of precipitation particles cannot be resolved explicitly by cloud resolving and climate models. Instead, they are represented by microphysics schemes that are based on a combination of theoretical knowledge, statistical assumptions, and fitting to data ("tuning"). Historically, tuning was done in an ad-hoc fashion, leading to parameter choices that are not explainable or repeatable. Recent work has treated it as an inverse problem that can be solved by Bayesian inference. The posterior distribution of the parameters given the data—the solution of Bayesian inference—is found through computationally expensive sampling methods, which require over O(10ˆ5) evaluations of the forward model; this is prohibitive for many models. We present a proof-of-concept of Bayesian learning applied to a new bulk microphysics scheme named "Cloudy", using the recently developed Calibrate-Emulate-Sample (CES) algorithm. Cloudy models collision-coalescence and collisional breakup of cloud droplets with an adjustable number of prognostic moments and with easily modifiable assumptions for the cloud droplet mass distribution and the collision kernel. The CES algorithm uses machine learning tools to accelerate Bayesian inference by reducing the number of forward evaluations needed to O(10ˆ2). It also exhibits a smoothing effect when forward evaluations are polluted by noise. In a suite of perfect-model experiments, we show that CES enables computationally efficient Bayesian inference of parameters in Cloudy from noisy observations of moments of the droplet mass distribution. In an additional imperfect-model experiment, a collision kernel parameter is successfully learned from output generated by a Lagrangian particle-based microphysics model.

Posted on 26 Nov 2022 — CC-BY-NC 4 — https://doi.org/10.1002/essoar.10510248.1 — This a preprint and has not been peer reviewed. Data may be preliminary.

An efficient Bayesian approach to learning droplet collision kernels: Proof of concept using "Cloudy", a new n-moment bulk microphysics scheme

Melanie Bieli[1], Oliver R. A. Dunbar[1], Emily K. de Jong[2], Anna Jaruga[1], Tapio Schneider[1], Tobias Bischoff[1]

[1]Division of Geological and Planetary Sciences, California Institute of Technology, Pasadena, CA, USA
[2]Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA, USA

Key Points:

- Historically, microphysics schemes were tuned to data in an ad-hoc way, resulting in parameter values that are not repeatable or explainable
- Bayesian inference puts uncertainty quantification and parameter learning on solid mathematical grounds, but is computationally expensive
- We present a proof-of-concept of computationally efficient Bayesian learning applied to a new bulk microphysics scheme called "Cloudy"

Corresponding author: Melanie Bieli, `melanie@caltech.edu`

Abstract

The small-scale microphysical processes governing the formation of precipitation particles cannot be resolved explicitly by cloud resolving and climate models. Instead, they are represented by microphysics schemes that are based on a combination of theoretical knowledge, statistical assumptions, and fitting to data ("tuning"). Historically, tuning was done in an ad-hoc fashion, leading to parameter choices that are not explainable or repeatable. Recent work has treated it as an inverse problem that can be solved by Bayesian inference. The posterior distribution of the parameters given the data—the solution of Bayesian inference—is found through computationally expensive sampling methods, which require over $\mathcal{O}(10^5)$ evaluations of the forward model; this is prohibitive for many models.

We present a proof-of-concept of Bayesian learning applied to a new bulk microphysics scheme named "Cloudy," using the recently developed Calibrate-Emulate-Sample (CES) algorithm. Cloudy models collision-coalescence and collisional breakup of cloud droplets with an adjustable number of prognostic moments and with easily modifiable assumptions for the cloud droplet mass distribution and the collision kernel. The CES algorithm uses machine learning tools to accelerate Bayesian inference by reducing the number of forward evaluations needed to $\mathcal{O}(10^2)$. It also exhibits a smoothing effect when forward evaluations are polluted by noise. In a suite of perfect-model experiments, we show that CES enables computationally efficient Bayesian inference of parameters in Cloudy from noisy observations of moments of the droplet mass distribution. In an additional imperfect-model experiment, a collision kernel parameter is successfully learned from output generated by a Lagrangian particle-based microphysics model.

Plain Language Summary

Clouds contain gazillions of cloud droplets, which grow by colliding and sticking together with each other, and eventually they become big enough to fall out as rain. Keeping track of every one of these droplet in weather and climate models is impossible, so the formation of rain has to be represented by simplified models, so-called "microphysics schemes". These schemes have become a bit like black boxes, with baked-in statistical assumptions and some empirical parameters whose values are somewhat obscure and not explainable. We show that we can use a method called Bayesian inference to determine the values of these parameters in a way that is both mathematically sound and reasonably fast. The idea of Bayesian inference is to come up with a first guess about the possible values of the parameters, and then to systematically refine that guess using observed data. We apply this method to a new microphysics scheme that we developed and named "Cloudy". To be honest, the data we use for our experiments are not real observations from, say, satellites, but are generated by Cloudy itself. With real observations, the problem becomes hairier, so what we do here is only a proof-of-concept — but hey, it's a start!

1 Introduction

Cloud microphysics comprises all processes controlling the formation and growth of cloud droplets and ice crystals and their fallout as precipitation. These processes play a key role in the climate system, affecting surface precipitation, latent heating and cooling, cloud radiative properties, and cloud chemistry. Due to the small scales on which they occur (sub-microns to centimeters), explicitly simulating the growth of individual cloud particles in a turbulent cloud requires model resolutions at least as small as the Kolmogorov scale, which is about 1 mm in the Earth's atmosphere. With horizontal grid spacings of about 10-50 km, state-of-the-art climate models are (and will remain) orders of magnitude too coarse to resolve the vast number of hydrometeors—typically about 10^8 in 1 m^3 of cloudy air—on a global scale.

Instead, global climate models (GCMs) represent microphysical processes by means of statistical parameterizations, which are developed based on a combination of physical understanding, statistical assumptions, heuristics, and tuning to observations. GCMs typically employ so-called bulk schemes, which assume some functional form of the cloud droplet size distribution (DSD) and step one or more statistical moments of the distribution forward in time. More computationally expensive representations of microphysics include bin schemes, which partition the DSD into discrete size bins and model the processes affecting the particles in each bin. More recent developments include Lagrangian particle-based schemes (e.g., Andrejczuk et al., 2010; Riechelmann et al., 2012; Shima et al., 2009, 2019), which simulate an ensemble of computational particles ("super-particles" or "super-droplets"), each representing a large number of real cloud and precipitation particles. The closest approximations to "ab initio" calculations of microphysical processes are performed by direct numerical simulations (DNS), which track the motion and growth of each individual cloud particle. To do that, they need to resolve the smallest scales of turbulence (millimeter and submillimeter scales), which limits typical domain sizes to less than 1 m^3.

Morrison, van Lier-Walqui, Fridlind, et al. (2020) provide an overview of the different approaches to the numerical modeling of microphysics and argue that the Lagrangian particle-based methods overcome several shortcomings of traditional bulk and bin schemes. A particularly attractive property of Lagrangian schemes is that in the limiting case where each computational particle represents a single real particle and the model resolution approaches that of a DNS, the Lagrangian scheme converges to the particle-by-particle DNS (Dziekan & Pawlowska, 2017). Due to their computational cost, though, Lagrangian particle-based methods will likely not replace the bulk schemes in global models within the next 1–2 decades. However, Lagrangian particle-based schemes can be used for cloud modeling and, as shown in this study, they can provide a benchmark for testing bulk schemes.

All three types of microphysics schemes (bulk, bin, and Lagrangian) rely on empirical parameters to compute process rates. Ultimately, this is a consequence of the fact that there is no known complete set of equations governing these microscopic processes, i.e., there is no microphysics analog to the Navier-Stokes equations. Given the limited theoretical knowledge, data and observations play a crucial role in constraining the values of empirical parameters. Historically, the process of determining ("tuning") these values has not been approached in a systematic and transparent way. However, a number of recent studies demonstrate the use of Bayesian techniques to parameter estimation in bulk microphysics schemes (e.g., Posselt & Vukicevic, 2010; Posselt, 2016; van Lier-Walqui et al., 2014). In a two-part paper, Morrison, van Lier-Walqui, Kumjian, and Prat (2020) introduce the Bayesian Observationally-constrained Statistical-physical Scheme (BOSS), a framework for the bulk parameterization of microphysics, which is designed to learn microphysical parameter distributions from data by means of Bayesian inference. The second part (van Lier-Walqui et al., 2020) gives a demonstration in the form of a perfect-model experiment, which shows that BOSS can be used in conjunction with a Markov chain Monte Carlo (MCMC) sampling algorithm to estimate parameters from synthetically generated rain observations. A key feature of BOSS is its adjustable complexity: While traditional schemes have fixed numbers of prognostic moments, BOSS allows the number of prognostic moments to be chosen flexibly, depending on the application and the observations available.

In a similar vein, to model parametric uncertainty with a strong mathematical foundation, we use a Bayesian framework, where model parameters are described by random variables. We propose a prior form of the distributions and refine them systematically with observed data, using a process known as Bayesian inference, Bayesian calibration, or uncertainty quantification. As in Schneider et al. (2017), we use the word "data" for any information source that is used as a ground truth against which a microphysics scheme is compared and calibrated, including both observations of natural clouds (e.g., satellite products or in-situ airborne observations) and output of higher-resolution or more physical model simulations. There is also much to be gained from using a perfect-model setting in a proof-of-concept, where the same model is used for both generating data and for inversion; however, the resulting parameter learning results will necessarily be optimistic. The standard tool of Bayesian inference is MCMC sampling, which represents this data-refined distribution empirically by providing a large set of samples drawn from it. The main drawback of MCMC methods is their computational cost: They require large numbers of model evaluations (typically about 10^5 to 10^6), which is not feasible for expensive models such as GCMs.

Recent work by Cleary et al. (2021) presents a method to perform an approximate Bayesian inversion of computationally expensive models for which derivatives are not readily available and whose evaluations may be polluted by noise, for example, from chaotic internal variability. This three-step method called Calibrate-Emulate-Sample (CES) has been shown to be effective for inferring parameter distributions in a convection scheme of a GCM exhibiting these properties (Dunbar et al., 2021; Howland et al., 2021). The calibrate step of the algorithm consists of ensemble Kalman inversion or variants such as the ensemble Kalman sampler (EKS; Garbuno-Inigo, Hoffmann, et al., 2020), which are used to find pairs of parameters and their respective model outputs, automatically focusing on a region of the parameter space that is likely to have produced the observed data. In the emulate step, a Gaussian process (Rasmussen & Williams, 2006) is trained on these parameter-output pairs and serves as a surrogate (emulator) of the original (expensive) forward model. In the sampling step, the fast-to-evaluate emulator is used in the likelihood of an MCMC algorithm to sample the posterior distribution in a computationally efficient manner.

The goal of this study is twofold: First, we introduce a new bulk microphysics framework that was designed for consistent representation of microphysical processes across models with different resolutions and physics. The model, called “Cloudy” (available at `https://github.com/CliMA/Cloudy.jl`), is broadly similar to BOSS (Morrison, van Lier-Walqui, Kumjian, & Prat, 2020; van Lier-Walqui et al., 2020), e.g., in that the number of prognostic moments is modifiable and that it can learn from data; however, there are also a few important differences, e.g., in that Cloudy allows for separate learning of collision kernels and DSDs, hence facilitating the finding of physically realizable solutions. Cloudy currently simulates collision-coalescence and collisional breakup of cloud droplets (with future development plans including an extension to other warm-rain processes and ice microphysics), in a way that the governing equations for the moments of the DSD can easily be related to the specific properties of collision kernels.

The second goal is to demonstrate that parameters in Cloudy can be learned from data in a computationally efficient way, through the approximate Bayesian inversion performed by CES. We present a suite of perfect-model experiments (where Cloudy itself is used to generate the data used for Bayesian inversion), as well as an experiment using data from simulations generated by PySDM (Bartman et al., 2021), a high-performance Python implementation of the super-droplet method (SDM) for representing liquid microphysics (package available at `https://github.com/atmos-cloud-sim-uj/PySDM`), with an additional process added to represent droplet breakup. In the spirit of a proof-of-concept, both Cloudy and PySDM are run in a computationally cheap zero-dimensional “box” framework.

This paper is organized as follows: Section 2 describes the underlying concepts and equations of Cloudy. In Section 3, we give a brief introduction to the Bayesian approach to solving inverse problems, together with an overview of the CES method. Section 4 explains the model experiments, including a summary of the PySDM model. The results of the experiments are shown in Section 5. The paper concludes with a summary of the findings in Section 6.

2 Cloudy Model Description

Cloudy is a flexible bulk microphysics model that simulates collision-coalescence and collisional breakup of cloud droplets. By "flexible," we mean the following:

- The number of prognostic moments can be adjusted to the requirements of the cloud droplet mass distribution function and availability of data for calibration.
- A modular design facilitates experimenting with different collision kernels and cloud droplet mass distribution functions.
- It is set up for Bayesian inference, i.e., parameters of collision kernels do not have to be fixed but instead can be learned from data.

The three main inputs required to run the model are: an initial droplet mass distribution function, a collision kernel specifying the rate of collisions between particles, and a coalescence efficiency defining the fraction of collisions that result in coalescence of the particles into one larger drop, as opposed to collisions that result in breakup of the particles into smaller fragments. Cloudy then simulates how the distribution (characterized by a set of n prognostic moments) evolves over time as a result of the droplet interactions defined by the given collision kernel and coalescence efficiency. The number of prognostic moments is determined ab initio by the type of the cloud droplet mass distribution to be simulated (e.g., a Gamma distribution or a mixture of Gamma distributions). Assuming a fixed distribution type is the central closure assumption made in Cloudy; the number of prognostic moments has to be chosen such that the distribution parameters can be computed from the prognostic moments. Note that not all distributions can be inferred uniquely from their moments; for example, a lognormal distribution is not uniquely defined by its moments because its moment generating function does not converge.

The mathematical core of the model consists of two equations: the stochastic collection equation (SCE; Smoluchowski, 1916) and the stochastic breakup equation (SBE; e.g., Pruppacher & Klett, 1978), both expressed in terms of the DSD moments. The SCE describes the time rate of change of $f = f(m,t)$, the mass distribution function of liquid water droplets, due to the process of collision and coalescence. The distribution f depends on droplet mass m and time t; generally, it will also depend on position in space, but we neglect this dependence in our zero-dimensional setting here. The mass distribution function is defined such that $f(m,t)\ \mathrm{d}m$ denotes the number of droplets with masses in the interval $[m, m+dm]$ per unit volume at time t. We will mostly refer to $f(m,t)$ by the term "particle mass distribution" (PMD) rather than by "droplet size distribution," even though the two expressions could be used interchangeably for spherical water droplets (the only type of droplet considered in this study), where there is a one-to-one map between droplet size and droplet mass. By deviating from the traditional terminology, we account for the possibility that a future version of Cloudy may include non-spherical and non-liquid particles.

Following Beheng (2010), the SCE can be written as

$$\left.\frac{\partial f(m,t)}{\partial t}\right|_{\mathrm{coal}} = \frac{1}{2}\int_{m'=0}^{\infty} f(m',t)f(m-m',t)C(m',m-m')\ \mathrm{d}m' - f(m,t)\int_{m'=0}^{\infty} f(m',t)C(m,m')\ \mathrm{d}m'. \tag{1}$$

The collection kernel $C(m',m'') = K(m',m'')E_c(m',m'')$ (units: cubic centimeters per particle per second) describes the rate at which two droplets of masses m' and m'' come into contact and coalesce. It is the product of the collision kernel $K(m',m'')$ and the dimensionless coalescence efficiency $E_c(m',m'')$, which denotes the fraction of droplets that coalesce into a drop of mass $m'+m''$ upon colliding with each other. Throughout this paper, we will assume E_c to have a constant value, even though in reality it depends on the kinetic energy of the two colliding droplets (e.g., Low & List, 1982; Beard & Ochs, 1995). The first term on the right-hand side of Eq. (1) describes the rate of increase of the number of drops having a mass m due to collision and coalescence of drops of masses m' and $m-m'$ (where the factor $\frac{1}{2}$ avoids double counting); the second term describes the rate of reduction of drops of mass m due to collision and coalescence of drops having a mass m with other drops.

The SBE describes the time evolution of the PMD due to collision-induced breakup and is given by

$$\left.\frac{\partial f(m,t)}{\partial t}\right|_{\text{breakup}} = \frac{1}{2}\int_{m'=0}^{\infty} f(m',t)\,\mathrm{d}m' \int_{m''=0}^{\infty} f(m'',t)B(m',m'')P(m;m',m'')\,\mathrm{d}m'' \\ - f(m,t)\int_{m''=0}^{\infty} \frac{f(m'',t)B(m',m'')}{m+m''}\,\mathrm{d}m'' \int_{m'=0}^{m+m''} m'P(m';m,m'')\,\mathrm{d}m'. \tag{2}$$

The breakup kernel $B(m',m'') = K(m',m'')(1-E_c(m',m''))$ (units: cubic centimeters per particle per second) defines the rate at which two droplets of masses m' and m'' come into contact and break apart. The function $P(m,m',m'')$ is the mass distribution function of the fragments m produced by collisional breakup of two droplets of masses m' and m'', with $P(m,m',m'')\,\mathrm{d}m$ giving the number of drops in the mass interval between m and $m+\mathrm{d}m$ resulting from the breakup. We use the exponential fragment distribution introduced by Feingold et al. (1988),

$$P(m;m',m'') = \nu^2(m'+m'')\exp(-\nu m), \tag{3}$$

where $\nu = (qM_0/M_1)$; M_0 (cm^{-3}) is the initial value of the zeroth PMD moment (i.e., the initial number of droplets), M_1 (g cm^{-3}) is the first PMD moment (i.e., the water content), and q is a positive integer characterizing the fragment concentration.

Conservation of mass dictates that the mass of the sum of all fragments must equal the mass of the two colliding drops, i.e.,

$$\int_0^{m'+m''} mP(m;,m',m'')\,\mathrm{d}m = m'+m''. \tag{4}$$

If the fragment distribution conserves mass exactly, the last integral in Eq. (2) evaluates to $m+m''$ and, after division by the denominator of the previous term, results in a multiplication by 1. However, the fragment distribution by Feingold et al. (1988) used in this study does not fulfill conservation of mass exactly, so the last integral in Eq. (2) cannot be omitted.

We will rewrite Eqs. (1) and (2) in terms of the moments M_k of $f(m,t)$, which are the prognostic microphysical variables in Cloudy. They are defined by

$$M_k = \int_0^{\infty} m^k f(m,t)\mathrm{d}m. \tag{5}$$

The time rate of change of the k-th moment of f is obtained by multiplying Eqs. (1) and (2) by m^k and integrating over the droplet mass, which yields

$$\begin{aligned}\left.\frac{\partial M_k}{\partial t}\right|_{\text{coal}} &= \frac{1}{2}\int_0^{\infty}\int_0^{\infty}\left((m+m')^k - m^k - m'^k\right)C(m,m')f(m,t)f(m',t)\,\mathrm{d}m\,\mathrm{d}m' \\ &=: I_{k,\text{ coal}}, \\ \left.\frac{\partial M_k}{\partial t}\right|_{\text{breakup}} &= \frac{1}{2}\int_0^{m'+m''} m^kP(m;m',m'')\mathrm{d}m\int_0^{\infty} f(m',t)\,\mathrm{d}m'\int_0^{\infty} f(m'',t)B(m',m'')\,\mathrm{d}m'' \\ &\quad - \int_0^{\infty} m^k f(m,t)\,\mathrm{d}m\int_0^{\infty}\frac{f(m'',t)B(m,m'')}{m+m''}\,\mathrm{d}m''\int_0^{m+m''} m'P(m';m,m'')\,\mathrm{d}m' \\ &=: I_{k,\text{ breakup}}.\end{aligned} \tag{6}$$

Thus, the time evolution of the kth moment M_k due to collision-coalescence and collisional breakup is given by

$$\frac{\partial M_k}{\partial t} = I_{k,\text{ coal}} + I_{k,\text{ breakup}} \tag{7}$$

To step the prognostic moments forward in time, Cloudy computes the right-hand side of Eq. (7) using Monte Carlo integration, a technique for numerical integration using random numbers. Compared with other numerical integration methods, Monte Carlo integration has the following advantages: It converges in any dimension, regardless of the smoothness of the integrand, albeit only at a rate of $\mathcal{O}(\sqrt{N})$ (where N is the number of random samples used to compute the integral). It is also conceptually simple and parallelizable. A further desirable feature is the fact that it adds a stochastic element to the simulations. The resulting internal model variability provides a straightforward way of estimating the covariance of the observational noise needed as input to the CES algorithm in a perfect-model setting (see section 3.2). With a deterministic integration method such as a numerical quadrature, there would not be any randomness in the simulated data, and artificial noise would have to be added to mimic observational noise.

The drawback of Monte Carlo integration is its computational cost: Many samples—thousands or even millions—may be required to obtain results of acceptable accuracy. Numerous techniques have been developed to reduce the variance of the Monte Carlo estimator, and hence the number of samples needed (e.g., Kleijnen & Rubinstein, 2013). For our specific application, 200 samples turned out to produce sufficiently accurate results (see Appendix A for implementation details of the Monte Carlo integration). Monte Carlo integration can easily be extended to higher dimensions and is therefore typically applied to compute high-dimensional integrals, but we use it here to demonstrate that CES works even with models that produce noisy output, as well as for its general-purpose functionality and robustness with respect to the integrand. For the simple model setup presented here, the computational cost of the Monte Carlo integration is easily affordable. The algorithm can even be applied in larger-scale settings as its embarrassingly parallel nature can be exploited very efficiently by GPUs (e.g., Kanzaki, 2011; Borowka et al., 2019).

Evaluating the integrals on the right-hand side of Eq. (7) requires knowledge of the PMD function $f(m,t)$. The initial distribution $f(m,t=0)$ is specified by the user. There is a priori no reason to assume that the PMD would retain its functional form as particles are colliding, are forming new drops, and are breaking apart. However, to uniquely identify the distribution f at each time step, one would have to keep track of infinitely many moments of the PMD, which is obviously not practicable. The truncation of this infinite system is the moment closure problem, which all moment-based microphysics schemes have to address in some form. In Cloudy, as in most bulk microphysics schemes, the closure is achieved by assuming an analytic functional form for the PMD, and allowing for the parameters of the PMD to change over time while the type of distribution itself is kept fixed. Thus, a Cloudy simulation consists of the following steps, which are summarized in Fig. 1:

1. The user specifies the collision kernel $K(x,y)$, initial PMD $f(m,t=0;\xi_0)$ with distribution parameters ξ_0, the coalescence efficiency E_c (assumed to be constant, as mentioned above), and the end time t_{end} of the simulation. In addition, a map $h_{\xi\to M}$ and a map $h^{-1}_{M\to\xi}$ have to be supplied. The former defines how to compute the prognostic PMD moments $M=\{M_k\}_{k=1,\dots,n}$ from the parameters ξ of the PMD, and the latter defines the inverse map from the prognostic moments to the parameters. Note that except for simple distributions, $h^{-1}_{M\to\xi}$ does not have a closed-form representation, and the PMD parameters have to be determined by solving an optimization problem.

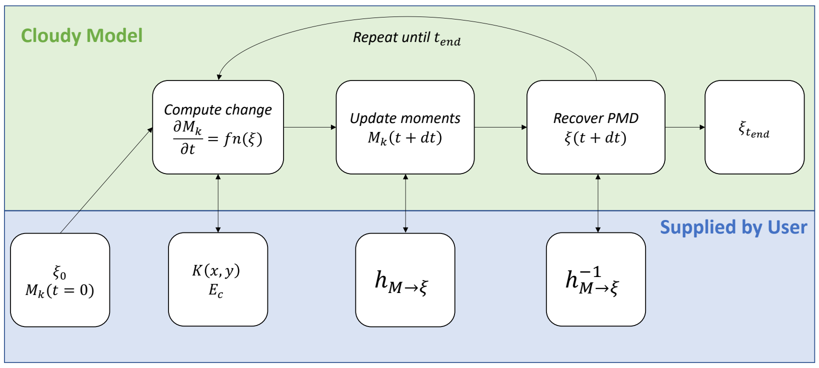

Figure 1. Summary of the computational steps performed by Cloudy and the required user-defined input. See text for notation.

2. The contributions of collision-coalescence and collisional breakup to the time evolution of each prognostic moment (right-hand side of Eq. (7); denoted $fn(\xi)$ in Fig. 1) is computed.
3. The prognostic moment are stepped forward in time.
4. The new parameters of the PMD are computed from the updated moments using $h^{-1}_{M\to\xi}$.
5. Steps 2–4 are repeated until t_{end} is reached.

One of the guiding principles in developing Cloudy was to make the scheme amenable to learning from data. Its modular design makes it easy to experiment with different kernels and PMDs, and the number of prognostic moments is determined by the user-provided parameter-to-moment map $h_{\xi\to M}$. We focus here on learning parameters of the collision kernel $K(x, y)$, though alternatively (and with slight modifications of the setup), Cloudy can be used to learn parameters of the PMD instead or in addition.

The approach to the closure problem is a notable difference between Cloudy and BOSS. In contrast to Cloudy, BOSS does not assume a functional form for the DSD. Instead, the diagnostic moments are expanded as multivariate generalized power series of the set of prognostic moments, and the DSD normalization method of Morrison et al. (2019) is used to relate the moments to one another statistically. While the assumption of a DSD form results in a loss of generality, it also allows for a clear separation of the parameters associated with the physics of collision-coalescence and breakup (as defined by the parameters of the collision kernel) from those associated with the droplet population (as defined by the parameters of the DSD). This improves the interpretability of the scheme and helps ensure physical realizability of the simulations.

3 Methods

3.1 Bayesian parameter estimation

The estimation of model parameters such as the coefficients of a collision-coalescence kernel can be formulated as a Bayesian inverse problem, whose solution—the posterior distribution of the unknown parameters given the observed data—is given by Bayes' rule:

$$f_{\theta|y}(\theta|y) = \frac{f_{y|\theta}(y|\theta) f_\theta(\theta)}{f_y(y)} \quad \propto \; f_{y|\theta}(y|\theta) f_\theta(\theta). \tag{8}$$

Here, $f_{\theta|y}(\theta|y)$ is the posterior probability density function (PDF) of the parameters θ given the data y, $f_{y|\theta}(y|\theta)$ is the likelihood function of the data given the parameters, $f_\theta(\theta)$ is the prior PDF of the parameters, and the normalization factor $f_y(y)$ is the marginal PDF of the data. In a Bayesian framework, the unknown parameters are thus modeled as random variables, and the posterior distribution can be written in terms of contributions from both prior information about the parameters and the likelihood of the observed data. Note that sampling the posterior distribution using MCMC methods does not require knowledge of the normalization factor $f_y(y)$.

We assume that the data y are linked to the parameter vector θ according to the additive relationship

$$\begin{aligned} y &= \mathcal{G}(\theta) + \eta_s \\ &= \overline{\mathcal{G}}(\theta) + \eta_y + \eta_s \\ &= \overline{\mathcal{G}}(\theta) + \eta. \end{aligned} \tag{9}$$

Here, the forward map $\mathcal{G} : \mathbb{R}^p \to \mathbb{R}^d$ maps a parameter vector $\theta \in \mathbb{R}^p$ to a d-dimensional output space; the error term η_s denotes structural error, which arises from a model's inability to accurately represent its target due to deficiencies in its mathematical structure. Because of the randomness introduced by the Monte Carlo integration (see section 2), the output of the forward map $\mathcal{G}$ is polluted by noise. We can think of $\mathcal{G}(\theta)$ as noisy observations of an underlying "true", deterministic forward map $\overline{\mathcal{G}}$: $\mathcal{G}(\theta) = \overline{\mathcal{G}}(\theta) + \eta_y$, where η_y is observational noise. The total error $\eta \in \mathbb{R}^d$ is thus the sum of two terms, observational noise η_y and structural error η_s, which are assumed to be independent. The choices we make for η_y and η_s will encode our assumptions about structure and origin of the error in a given scenario.

The parameter-to-data map $\mathcal{G}$ consists of two components. The main component is the map $\Psi : \Omega \to \mathbb{R}^d$, involving a forward simulation of Cloudy using parameters sampled from a physical parameter domain Ω. The map Ψ can be thought of as a dynamical model, whose output depends on p model parameters that we wish to learn (here, the model parameters to be learned are parameters of collision kernels). So that computational methods interface only with unbounded distributions, we choose to work always with unbounded parameter distributions θ in tandem with an invertible transformation function $\mathcal{T}\colon \Omega \to \mathbb{R}^p$. The combined map $\mathcal{G}$ from parameters to data thus takes the form

$$\mathcal{G} = \Psi \circ \mathcal{T}^{-1}. \tag{10}$$

Sampling the posterior distribution (Eq. 8) requires the repeated evaluation of the data likelihood, whose distribution is given by

$$f_{y|\theta}(y|\theta) = f_\eta(y - \overline{\mathcal{G}}(\theta)), \tag{11}$$

where f_η is the PDF of the noise η. Initially, only the noisy map $\mathcal{G}$ is available, such that the likelihood given by Eq. (11) cannot be evaluated. The underlying true model $\overline{\mathcal{G}}(\theta)$ only becomes accessible through the training of a Gaussian process emulator in the middle stage of CES (see section 3.2 and Dunbar et al., 2021)—this emulator learns a smooth function which is a surrogate for $\overline{\mathcal{G}}(\theta)$.

We work with two different model-data scenarios. In a first scenario, we consider a perfect-model experiment where we assume that there is no structural uncertainty, $\eta_s = 0$. We also choose η_y to be realizations of random noise due to measurement error. When using data y that are statistical aggregates such as moments of the PMD or other averages, and assuming a perfect (unbiased) model, it is reasonable to use a central-limit

Table 1. Overview of collision kernels, kernel parameters, their prior distributions and constraints. For any element X of a parameter θ living in the unconstrained space, the mapping to the corresponding parameter value x in the constrained space with a uniform prior is given approximately by the transformation $\mathcal{T}^{-1}(X) = (x_{\text{upper}} \exp(X) + x_{\text{lower}})/(\exp(X) + 1)$.

Kernel	Name	Unconstrained Parameter	Prior	Constrained Parameter	Constraint
$K(x,y) = b(x+y)$	Sum-of-masses	$\theta = [B]$	$\theta \sim \mathcal{N}(0,1)$	$\phi = [b]$	$[b_{\text{lower}}, b_{\text{upper}}] = [10^2, 10^4]$ cm^3 g^{-1} s^{-1}
$K(x,y) = a + b(x+y)$	Sum-of-masses plus constant	$\theta = [A, B]$	$\theta \sim \mathcal{N}(0,1)$	$\phi = [a, b]$	$[a_{\text{lower}}, a_{\text{upper}}] = [10^{-7}, 10^{-5}]$ cm^3 s^{-1} $[b_{\text{lower}}, b_{\text{upper}}] = [10^2, 10^4]$ cm^3 g^{-1} s^{-1}
$K(x,y) = \begin{cases} b(x+y) & x < c \text{ or } y < c \\ a & \text{otherwise} \end{cases}$	Piecewise	$\theta = [A, B, C]$	$\theta \sim \mathcal{N}(0,1)$	$\phi = [a, b, c]$	$[a_{\text{lower}}, a_{\text{upper}}] = [10^{-7}, 10^{-5}]$ cm^3 s^{-1} $[b_{\text{lower}}, b_{\text{upper}}] = [10^2, 10^4]$ cm^3 g^{-1} s^{-1} $[c_{\text{lower}}, c_{\text{upper}}] = [5.0 \times 10^{-11}, 5.0 \times 10^{-9}]$ g

theorem, so that η_y is a draw from a (multivariate) normal random variable with zero mean and covariance matrix Γ_y.

For each experiment in this scenario, we approximate the observational covariance by running the forward model 10 times with true parameter values and estimating Γ_y as the sample covariance matrix of the resulting data, which are 3-dimensional vectors of PMD moments (note that in order to give a non-singular estimate of the covariance matrix, the number of samples needs to be greater than the dimensionality of the data). The data y are taken to be the sample mean of these 10 vectors of PMD moments.

In a second scenario, the data y are generated by PySDM (see section 4) instead of Cloudy. Structural uncertainties arise from the differences between the modeling approaches of Cloudy and PySDM (bulk vs. Lagrangian particle-based), with PySDM arguably simulating droplet-droplet interactions in a more realistic way (e.g., due to the lack of a closure assumption).

Here, we opt for a simple representation of the model discrepancy η_s as a (multivariate) normal random variable with constant mean $m_s \in \mathbb{R}^d$ and covariance Γ_s. Explicitly, we write this as

$$y = \mathcal{G}(\theta) + m_s + \tilde{\eta}, \qquad \text{where } \tilde{\eta} \sim N(0, \Gamma_y + \Gamma_s) \tag{12}$$

As both PySDM and Cloudy have the same model parameters, we can estimate the bias m_s by running each model ten times with at the true parameter value and taking the difference of the sample means of the resulting output. Apart from the addition of a bias term, Eq. (12) also differs from the perfect-model version (Eq. 9) in that the covariance of the noise $\tilde{\eta}$ is the sum of the Cloudy and PySDM noise covariance matrices.

In general, assessing a model's adequacy to reproduce the given data (even when the model is perfectly calibrated) is a difficult task (e.g., Brockwell & Davis, 1996; Kennedy & O'Hagan, 2001; Weisberg, 2014). Our choice can be seen as a special case of the approach followed by Kennedy and O'Hagan (2001), where η_s is modeled with a Gaussian process.

3.2 Calibrate-Emulate-Sample

The Calibrate-Emulate-Sample (CES) method (Cleary et al., 2021) is designed for Bayesian inversion in settings where the forward model is too computationally expensive and/or noisy for direct sampling of the posterior using, e.g., MCMC. We give a conceptual overview of CES and refer to Cleary et al. (2021) and Dunbar et al. (2021) for more detailed descriptions. The method accelerates Bayesian learning by substantially reducing the number of forward model evaluations required from the $\mathcal{O}(10^5)$–$\mathcal{O}(10^6)$ evaluations typically needed for MCMC to $\mathcal{O}(10^2)$ evaluations. It consists of three stages:

- The **calibration** stage uses ensemble Kalman inversion (EKI; Iglesias et al., 2013) or variants thereof such as ensemble Kalman sampling (EKS; Garbuno-Inigo, Hoffmann, et al., 2020; Garbuno-Inigo, Nüsken, & Reich, 2020) to solve the experimental design problem of choosing good training points for the subsequent emulation stage. EKI and EKS are derivative-free methods that place training points of the parameter-to-data map in the vicinity of where the Bayesian posterior distribution of the parameters is concentrated. They are highly parallelizable, scale well to high-dimensional problems, and are well suited to dealing with noisy forward model evaluations (Duncan et al., 2021). In this study, we use EKS, whose ensemble approximates the Bayesian posterior.
- In the **emulation** stage, the samples from the calibration stage are used to train a Gaussian process regression model, which serves as an emulator that approximates the original parameter-to-data map but is much cheaper to evaluate.
- The **sampling** stage uses MCMC methods to sample the posterior distribution of the parameters, using the cheap emulator instead of the original (expensive) forward model.

Cloudy is very cheap to run in the prototype setting of this paper; thus, CES is not necessary from the perspective of computational cost here. But we apply the method with an eye toward larger-scale applications in GCMs later, and for the smoothing property of the Gaussian process emulator, which increases robustness with respect to the noise induced by the Monte Carlo integration (see section 2 and Dunbar et al. (2021)). A proof-of-concept with a cheap model also has the advantage that results can be compared against traditional techniques of Bayesian inversion (namely, direct MCMC sampling without an emulator), which would not be feasible with a more computationally expensive model.

4 Experimental Setup

The goal of the experiments is to demonstrate that kernel parameters in Cloudy can be learned from data using the CES method. For this purpose, parameter learning is performed in a perfect-model setting, where the data are generated by running Cloudy with the "true" parameter values that are then to be learned by Bayesian inversion. This setup tests if the true parameters are identifiable in the absence of model uncertainty, i.e., in a scenario where the constrained model, with the correct parameters, is able to reproduce the data to within noise. In all experiments, the data are values of the zeroth, first, and second moment of the PMD at the end time t_{end} of the simulation. The zeroth moment of the PMD is equal to the total number concentration, while the second and third moments are proportional to the mass mixing ratio and radar reflectivity factor, respectively. Thus, the first three PMD moments are directly related to quantities that can be in principle obtained from remote sensing systems.

We also present an experiment where the data are generated by PySDM (Bartman et al., 2021), a Lagrangian super-droplet scheme based on the Monte Carlo algorithm by Shima et al. (2009), which models collisional growth of cloud droplets without using the stochastic collection equation. Instead, it represents the cloud droplet population by a number of computational super-droplets, each corresponding to some multitude of real droplets with identical properties (including size and position). The collision and coalescence of these super-droplets is modeled stochastically: Within each time step, only a discrete sample of super-droplet pairs is considered. This is done to reduce the computational cost from $\mathcal{O}(N_s^2)$, which would result from considering all pairs, to $\mathcal{O}(N_s)$, where N_s is the number of super-droplets. Each of these candidate pairs then collides with a probability that depends on the multiplicities of the two colliding super-droplets, i.e., on the numbers of real droplets they represent. A comprehensive description of the method is given in Shima et al. (2009), and its implementation in PySDM is detailed in Bartman et al. (2021).

For the purpose of this paper, a simple breakup implementation was added to PySDM. In this implementation, described in Appendix B, a breakup results in exactly two fragments, each of which carries half of the sum of the masses of the two colliding drops. When learning from data generated by PySDM, the same breakup implementation is used in Cloudy, by defining a fragment distribution

$$P(m, m', m'') = 2\delta\left(m - \frac{m' + m''}{2}\right), \tag{13}$$

where δ is the Dirac delta function.

We will present results of the following experiments (see also Table 1):

- Collision kernel of the form $K(x, y) = b(x+y)$: The parameter b is learned in a perfect-model setting, for $b = 2000$ cm^3 g^{-1} s^{-1}, 4000 cm^3 g^{-1} s^{-1}, 6000 cm^3 g^{-1} s^{-1}, and 8000 cm^3 g^{-1} s^{-1}. In addition to CES, two of these four experiments ($b =$ 2000 cm^3 g^{-1} s^{-1} and 4000 cm^3 g^{-1} s^{-1}) are also carried out with a brute-force MCMC sampling, to compare results and performance of the two methods. As the name suggests, brute-force MCMC sampling involves repeatedly (10^5 times) evaluating Cloudy itself rather than using the predictions of an emulator.
- Collision kernel of the form $K(x, y) = b(x+y)$: The parameter $b = 2000$ cm^3 g^{-1} s^{-1} is learned from data generated by PySDM, using CES.
- Collision kernel of the form $K(x, y) = a+b(x+y)$: The parameters a and b are learned in a perfect-model setting using CES, for $a = 4.0 \times 10^{-6}$ cm^{3}s^{-1} and $b = 3000$ cm^{3}g^{-1}s^{-1}.
- Collision kernel of the form

$$K(x, y) = \begin{cases} b(x + y) & x < c \text{ or } y < c, \\ a & \text{otherwise.} \end{cases} \tag{14}$$

The parameters a, b, and c are learned in a perfect-model setting using CES, for $a = 2.0 \times 10^{-6}$ cm^3 s^{-1}, $b = 3000$ cm^3 g^{-1} s^{-1}, $c = 1.0 \times 10^{-9}$ g.

The kernels chosen for this suite of experiments represent a sequence of increasingly difficult learning tasks (from learning one parameter to learning three parameters) that are used to assess the ability of the CES method to retrieve parameters and provide uncertainty quantification. How realistically these kernels represent droplet-droplet interactions from a physical perspective is of lesser concern for this purpose, but they are nevertheless inspired by established kernels: The sum-of-masses kernel $K(x, y) = b(x+y)$ is known as a Golovin kernel (Golovin, 1963), while the piecewise defined kernel is a simpler variant of a Long (1974) kernel, which is quadratic for small droplets and linear for large droplets[1].

The PMD is assumed to be a Gamma distribution parameterized by $\xi = [N_0, \alpha, \beta]$, where N_0 is a scaling constant (corresponding to the total number of droplets), α is the shape parameter, and β is the rate parameter:

$$f(m, t) = \frac{N_0 \beta^\alpha}{\Gamma(\alpha)} m^{\alpha-1} \exp(-\beta m). \tag{15}$$

[1] To be precise, both the Golovin kernel and the Long kernel are *collection* kernels, i.e., they represent the product of a collision kernel and the coalescence efficiency. This study is concerned with learning *collision* kernels, but since the coalescence efficiency is simply assumed to be constant, the resulting collection kernel is proportional to the underlying collision kernel.

The parameter vector ξ changes over time, as the shape of the distribution evolves. For Gamma mass distribution functions, specifying the first three moments is sufficient to uniquely determine the distribution parameters ξ, hence Cloudy solves Eq. (6) for $k = 0, 1, 2$ (but the number of prognostic moments can be adjusted to the requirements of any given closure distribution). The map $h_{\xi\to m}$ from the distribution parameters ξ to the PMD moments $M = [M_0, M_1, M_2]$ and its inverse $h^{-1}_{M\to\xi}$ are given by

$$
\begin{aligned}
h_{\xi\to M}(N_0,\alpha,\beta) &= \left[N_0, \frac{N_0\alpha}{\beta}, \frac{N_0\alpha(\alpha+1)}{\beta^2}\right] = [M_0, M_1, M_2] = M,\\
h^{-1}_{M\to\xi}(M_0,M_1,M_2) &= \left[M_0, \frac{1}{\frac{M_0M_2}{M_1^2}-1}, \frac{M_1M_0}{M_0M_2-M_1^2}\right] = [N_0,\alpha,\beta] = \xi.
\end{aligned}
\tag{16}
$$

The definition of $h^{-1}_{M\to\xi}$ shows that α goes to zero when the product of M_0 and M_2 increases over time (M_1 is approximately constant and does not cause much variation in the value of α). Small values of α lead to instabilities in Cloudy and eventually cause it to crash. The reason is that in the regime of $\alpha \approx 0$, the small changes in the prognostic moments that the adaptive time stepping produces correspond to large changes in the underlying distribution parameters. In addition, sampling from Gamma distributions (which is done in the Monte Carlo approximation of the coalescence and breakup integrals described in Appendix A) becomes inaccurate and inefficient when the shape parameter is small (e.g., Best, 1983; Liu et al., 2017). Collision-coalescence decreases M_0 and tends to increase M_2, while collisional breakup increases M_0 and tends to decrease M_2. The combined effect can be such that M_0 grows faster than M_2 declines (or vice visa), resulting in an increasing product of the two moments. For the experiments presented in this paper, we circumvented this problem by choosing the settings (constraints for the kernel parameters, duration of the simulations, initial condition, value of the coalescence efficiency) such that the resulting simulations were stable.

The simulations are initialized with 10^4 particles with a mean mass and standard deviation 0.33×10^{-9} g, corresponding to initial moments $M_0 = 10^4$ cm^{-3}, $M_1 = 3.30\times 10^{-6}$ g cm^{-3}, and $M_2 = 2.18\times10^{-15}$ g^2 cm^{-3}. They are run for a simulation time period of 60 s.

5 Results

5.1 Evolution of the PMD moments

Unless the coalescence efficiency is set to 0 or 1, the time evolution of the PMD is the result of two competing processes: Collision-coalescence reduces the number of droplets (decreasing M_0) and creates larger drops (increasing M_2), while collisional breakup generates more but smaller droplets (increasing M_0 and decreasing M_2). Both processes conserve the liquid water mass (M_1). To illustrate the effects of differences in the relative strength of these two processes, Fig. 2 shows the time evolution of M_0, M_1, and M_2 for coalescence efficiencies of 0.8, 0.9, and 1.0 (with a coalescence efficiency of 1.0, there is no breakup, i.e., all collisions result in coalescence). For each value of the coalescence efficiency in Fig. 2a, 10 simulations with identical collision kernels are run; the resulting spread in the moment evolution is due to the randomness inherent in the Monte Carlo integration used to compute Eq. (7). As is to be expected, the number of droplets in the simulation without breakup decreases monotonically, moving toward its theoretical limit of a single drop containing all the liquid water mass. With a coalescence efficiency of 0.9, the droplet number decreases more slowly over the time period shown, and with a coalescence efficiency of 0.8, breakup produces more droplets than coalescence removes. Mass is conserved most exactly for the no-breakup simulation, where determining the time rate of change of the moments does not involve the computation of the breakup term (Eq. 6). For coalescence efficiencies of 0.8 and 0.9, mass is conserved to within 6% of the initial mass. These deviations from mass conservation are due to the Monte Carlo approxima-

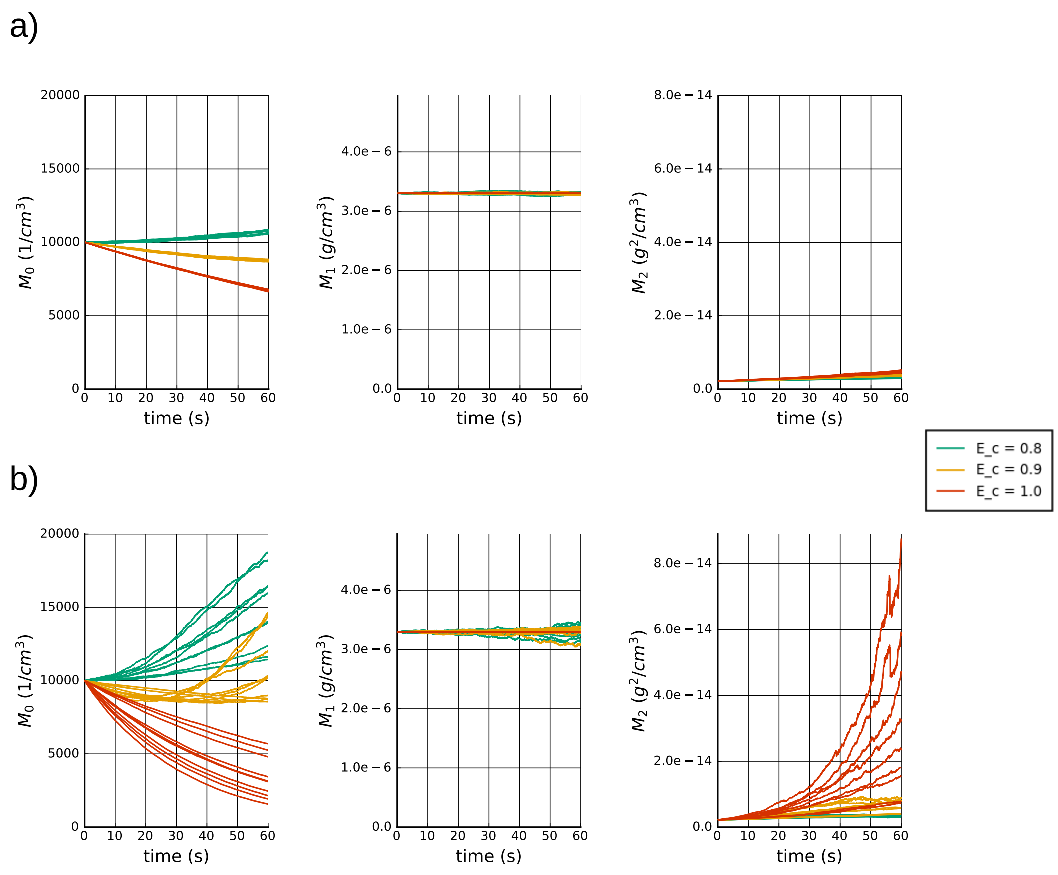

Figure 2. Evolution of the (left) zeroth, (middle) first, and (right) second moment of the PMD, for different values of the coalescence efficiency E_c (0.8, 0.9, and 1.0). All simulations use a kernel of the form $K(x, y) = b(x + y)$. (a) Simulations for each value of E_c are repeated 10 times with the same kernel ($b = 2000\ cm^3 g^{-1} s^{-1}$). (b) Simulations for each value of E_c are also repeated 10 times, but each time with a kernel parameter that is randomly drawn from its prior distribution.

tion to the coalescence and breakup integrals (Eq. 6) and due to numerical time stepping errors.

In Fig. 2b, each simulation is performed with a new kernel parameter b drawn from its prior distribution, such that the observed spread in the moment evolution is due to the combined randomness of sampling the kernel parameter and of the Monte Carlo integration. The former clearly accounts for a greater share of the variability, as can be seen by comparing Figs. 2b and Fig. 2a (note that both figures use the same y axes). Figure 2b also shows that the bounds on the kernel parameters (Table 1) are large enough for the M_0 evolution of different coalescence efficiencies to overlap in some cases.

5.2 "Sum-of-Masses" Kernel

Figure 3a shows the posterior distributions generated by the CES method for four different values of the parameter b in a sum-of-masses kernel $K(x, y) = b(x + y)$. The results are shown in the transformed, "unconstrained" space where the CES algorithm takes place and where the prior distribution of the parameter vector θ is defined. In Fig. 3b, the same results are shown in the constrained space where the model input lives. Note that our discussion of the posterior distributions is based on Fig. 3a, and all following results will be displayed only in the unconstrained space.

In all four experiments, the maximum a posteriori estimate is a good estimate of the true parameter value. The narrowest of the four distributions and hence the most certain parameter estimate is obtained for the smallest parameter value ($b = 2000$ cm^3 g^{-1} s^{-1}), while the largest of the four parameter values ($b = 8000$ cm^3 g^{-1} s^{-1}) results in the distribution with the largest spread. The distribution spread reflects the underlying noise in the data, which varies with the magnitude of the kernel parameter: the larger its value, the larger (in absolute value) the error in the Monte Carlo estimate of the coalescence and breakup integrals describing the time rates of change of the distribution moments (Eq. 6), and hence the larger the resulting variance in the data. This effect gets multiplied because the adaptive time stepper uses smaller step sizes when the solution is changing fast, leading to more evaluations of the coalescence and breakup integrals over the course of a simulation (about 10 times more evaluations for $b = 8000$ cm^3 g^{-1} s^{-1} than for $b = 2000$ cm^3 g^{-1} s^{-1}).

While the Gaussian approximation obtained from the ensemble mean and covariance of the last EKS iteration is a good approximation of the posterior distributions for $b = 4000$ cm^3 g^{-1} s^{-1} and $b = 6000$ cm^3 g^{-1} s^{-1}, it underestimates the mass in the tails of the posterior for $b = 2000$ cm^3 g^{-1} s^{-1} and does not capture the more cusp-like shape of the posterior for $b = 8000$ cm^3 g^{-1} s^{-1}. In all practical applications, the shape of the posterior is (by definition) unknown a priori and may differ substantially from the Gaussian approximation obtained in the calibration stage. Accurate uncertainty quantification thus requires sampling the posterior.

The results of these four experiments show that the CES method is able to retrieve the optimal parameter and provide uncertainty quantification in a perfect-model setting, for a one-parameter kernel. For comparison, Fig. 3 also shows the posterior distributions obtained from brute-force MCMC sampling without the calibration and emulation stages, which is about 10^3 times slower than CES. The similar shapes of the posterior distributions from these two methods confirm that CES produces a high-quality approximation to the true solution of this problem. Brute-force sampling was only possible for the two smaller parameter values; the higher noise levels in the simulations with the larger parameters caused the MCMC algorithm to get stuck in local maxima of the objective function. While there are advanced Monte Carlo methods such as simulated annealing (Kirkpatrick et al., 1983) that are less susceptible to local trapping, CES has the advantage of performing well even with simple MCMC implementations, thanks to the smoothing property of the GP emulator.

Learning the parameter of a sum-of-masses kernel from data generated by PySDM results in the posterior distribution shown in Fig. 4. As described in section 3.1, this experiment differs from the perfect-model experiments in that its underlying equation includes a bias term representing the model discrepancy, and an inflated noise representing the combined stochasticity of Cloudy and PySDM (Eq. 12). CES is able to provide uncertainty quantification and a good estimate of the true parameter value in this modified setup. Note however that in the absence of the "gold standard" posterior distribution obtained from brute-force MCMC sampling, the quality of the uncertainty quantification is not easily measured. This underlines the importance of testing any approach to uncertainty quantification in a setting that allows for comparison of the resulting posterior distribution with that obtained from a method such as MCMC, which provably converges to the desired posterior distribution (e.g., Robert & Casella, 2005).

5.3 "Sum-of-Masses Plus Constant" Kernel

We will visualize the output of each of the three stages of the CES algorithm using the example of the "sum-of-masses plus constant" kernel $K(x, y) = a + b(x + y)$ with $a = 4 \times 10^{-6}$ cm^3 s^{-1} and $b = 3000$ cm^3 g^{-1} s^{-1}.

In the first stage (Fig. 5), the EKS algorithm transforms an initial ensemble of $J = 50$ members through successive updates into approximate samples of the posterior distribution. The initial ensemble, randomly drawn from the relatively uninformative prior distributions of the parameters, is spread broadly over the parameter space. Over the course of subsequent iterations (each of which requires 50 model evaluations), the ensemble becomes concentrated near the true parameter values, with the sample mean and covariance of the ensemble sampled from a Gaussian approximation of the posterior distribution. This is a difference to EKI (Iglesias et al., 2013), a closely related optimization method whose iterative updates result in a collapse of the ensemble onto the optimal parameter. EKS produces better training points for the emulator but in its present forms usually requires more iterations.

The calibration stage generates $N_{it} \times J = 500$ parameter-data pairs, which are used to train the emulator. The Gaussian process emulator predicts the mean and the variance at any data point in its input space, conditional on the training data (Fig. 6). Thanks to the well chosen training points, which are concentrated around the mean or mode of the posterior distribution, the predictions are most confident (have smallest variance) near the optimal parameter (around $[A, B] = [-0.43, -0.88]$), i.e., near the minimum of the objective function.

The MCMC algorithm samples the posterior distribution using the predictions of the emulator instead of actual model evaluations. Figure 7 shows kernel density estimates of the MCMC results, with contours containing 5%, 10%, 50%, 75%, 90%, and 99% of the posterior mass. The true value (blue dot) is captured within the 5% contour of the posterior density, showing that the maximum a posteriori estimate of the parameters obtained by the CES method is a good approximation of the true optimum. Both the mean and shape of the MCMC sampled distribution differ from the distribution of the last EKS ensemble (red dots in Figure 5). Since EKS relies on a Gaussian assumption for the posterior distribution, its output may diverge from the true posterior when that assumption does not hold.

5.4 Piecewise Kernel

The piecewise kernel represents a scenario where the collision rates of droplets smaller than some (not precisely known) threshold differ from those of droplets exceeding that threshold. Figure 8 shows the posterior distributions for the three parameters of this kernel, together with their true values and prior distributions. The difference between the prior and posterior distributions reflects the information about the kernel parameters learned from the data. In this example, the largest information gain was achieved for the linear rate b, whose posterior shows the smallest uncertainty. However, the information gain from the prior to the posterior of the other two parameters is smaller than that of previous examples, especially for the mass threshold (parameter c). This reflects the difficulty of finding data that provide the necessary constraints on all three parameters of the piecewise kernel, and that make the inverse problem sufficiently well posed. For example, if the mass threshold c is not exceeded during a model run, the resulting output will only be sensitive to the linear rate b. Similarly, the information content of the model output is limited when the mass threshold is too small, when the effect of a is much bigger than that of b (or vice versa), etc. Which parameter can best be retrieved depends on the choice of true parameter values and their constraints, which has to ensure that the model output produced in the calibration stage is sensitive to all under-

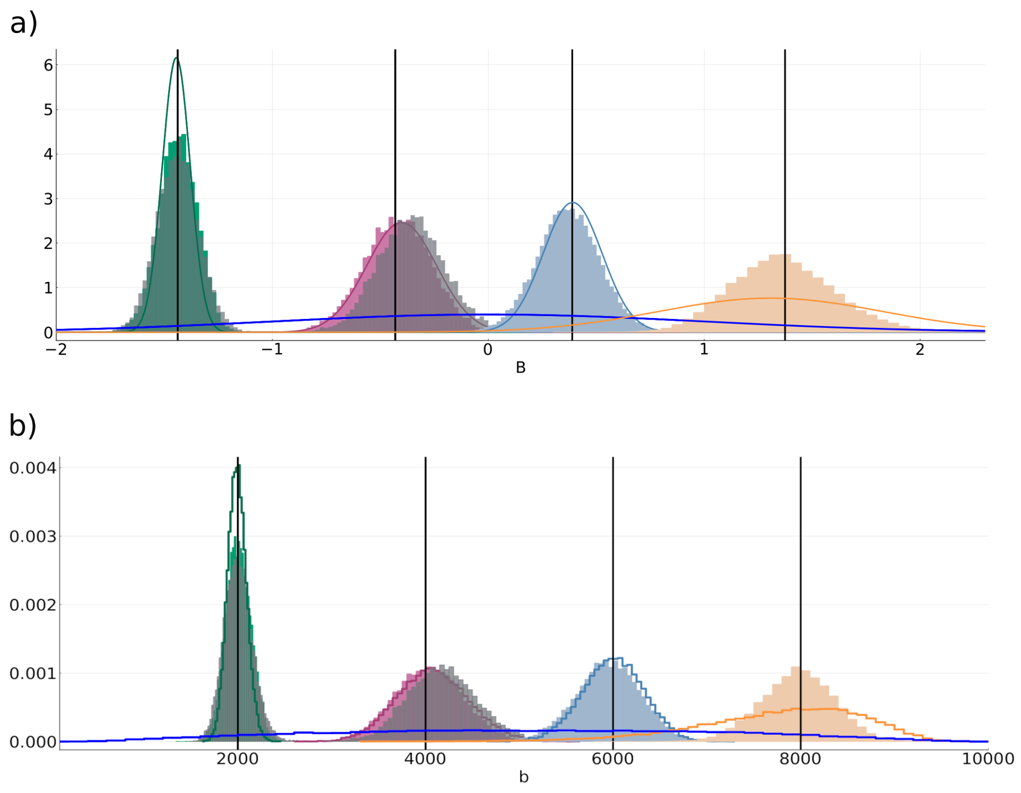

Figure 3. Histograms of MCMC samples of the posterior distributions obtained by the CES algorithm, for the inverse problem of finding the parameter b of a “sum-of-masses” kernel $K(x, y) = b(x + y)$. The results in (a) are shown in the unbounded space where the CES algorithm is performed; in (b), the same results are shown in the bounded space where the model input lives. Different colors correspond to different values of the true parameter b, each of which is marked by a vertical black line (from left to right, in the bounded space: $b = 2000$ cm^3 g^{-1} s^{-1}, $b = 4000$ cm^3 g^{-1} s^{-1}, $b = 6000$ cm^3 g^{-1} s^{-1}, $b = 8000$ cm^3 g^{-1} s^{-1}). The lines show Gaussian approximations to the posterior distributions, which are specified by the ensemble mean and standard deviation of the parameters in the last EKS iteration. All four experiments have the same prior parameter distribution shown as the dark blue line. The two additional histograms in grey (for $b = 2000$ cm^3 g^{-1} s^{-1} and $b = 4000$ cm^3 g^{-1} s^{-1}) show the results of brute-force MCMC sampling.

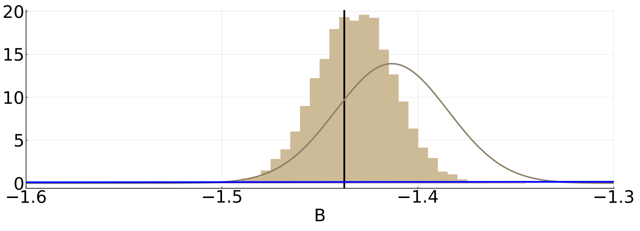

Figure 4. Histogram of MCMC samples produced by CES, showing the posterior distribution of the parameter b of a "sum-of-masses" kernel $K(x, y) = b(x + y)$, given data y generated by PySDM. The Gaussian approximation to the posterior distributions shown as a line is specified by the ensemble mean and standard deviation of the parameters in the last EKS iteration. The prior distribution of the parameter is shown in blue, and the true parameter value ($b = 2000$ cm^3 g^{-1} s^{-1} in the bounded space where the model input lives) is marked by a vertical black line. The plot is shown in the unbounded parameter space.

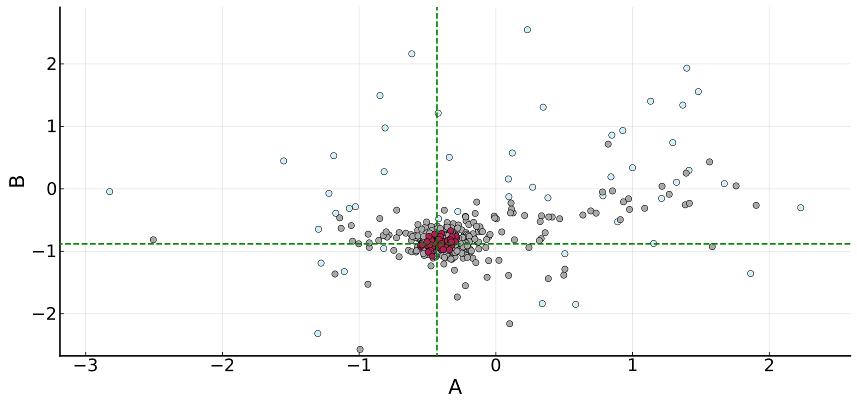

Figure 5. Evolution of the ensemble over subsequent EKS iterations, for the "sum-of-masses plus constant" kernel $K(x, y) = a + b(x + y)$ with $a = 4 \times 10^{-6}$ cm^3 s^{-1} and $b = 3000$ cm^3 g^{-1} s^{-1}. The initial ensemble (iteration 0) is highlighted in light blue; the subsequent eight iterations are colored in gray; and the final ensemble (iteration 9) is highlighted in red. The intersection of the dashed green lines represents the true parameter values used to generate observational data. The plot is shown in the unbounded parameter space.

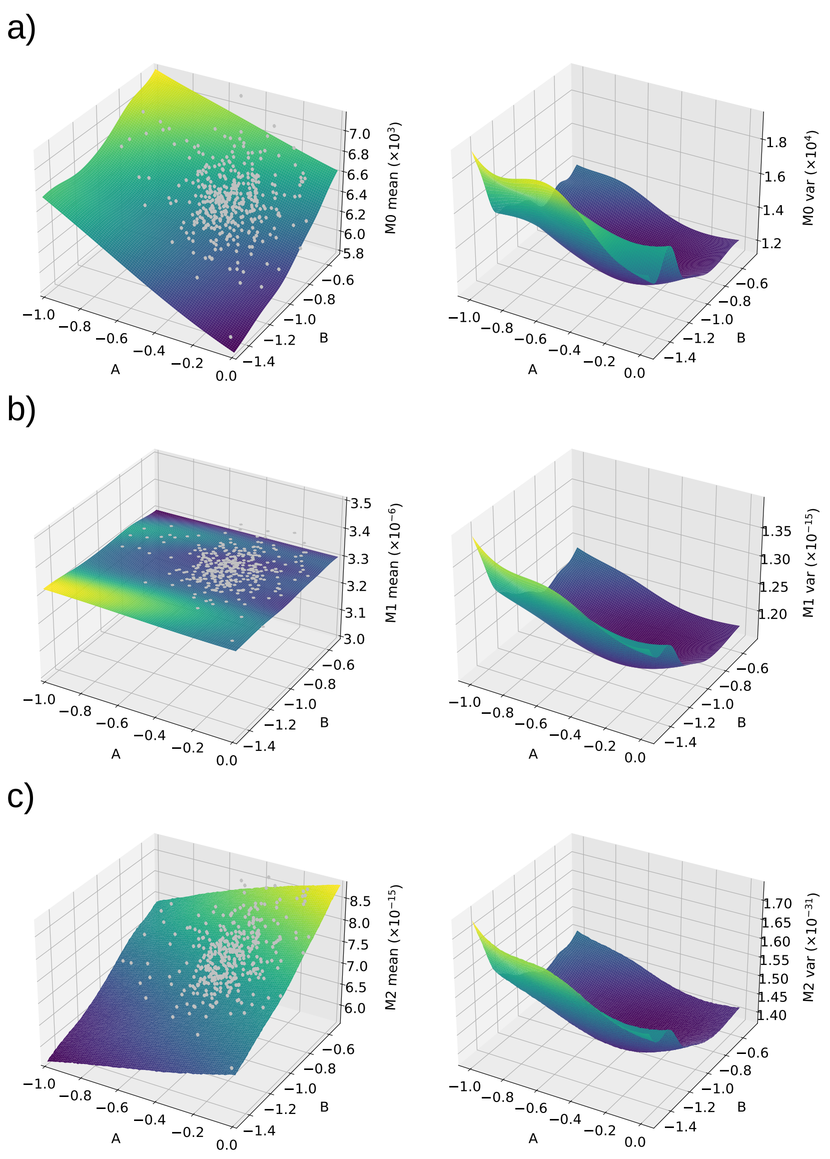

Figure 6. Predictions of the Gaussian process emulator for the "sum-of-masses plus constant" kernel $K(x, y) = a + b(x + y)$ with $a = 4 \times 10^{-6}$ cm^3 s^{-1} and $b = 3000$ cm^3 g^{-1} s^{-1}: (left) predicted mean and (right) predicted variance, for (a) M_0, (b) M_1, and (c) M_2. The grey dots represent training points generated during the calibration stage of CES (there are additional ones that fall outside the plotting domain). The predictions are shown in the unbounded parameter space.

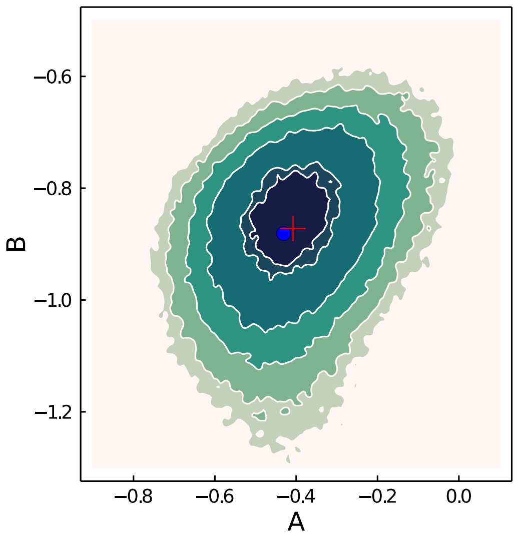

Figure 7. Density plot of MCMC samples of the posterior distribution, for the "sum-of-masses plus constant" kernel $K(x, y) = a + b(x + y)$ with $a = 4 \times 10^{-6}\text{cm}^3\ \text{s}^{-1}$ and $b = 3000\ \text{cm}^3\ \text{g}^{-1}\ \text{s}^{-1}$. The contours are contain 5%, 10%, 50%, 75%, 90% and 95% of the sampled distribution. The blue dot marks the true parameters, and the red cross is the average across ensemble members in the last EKS iteration. The posterior densities are shown in the unbounded parameter space.

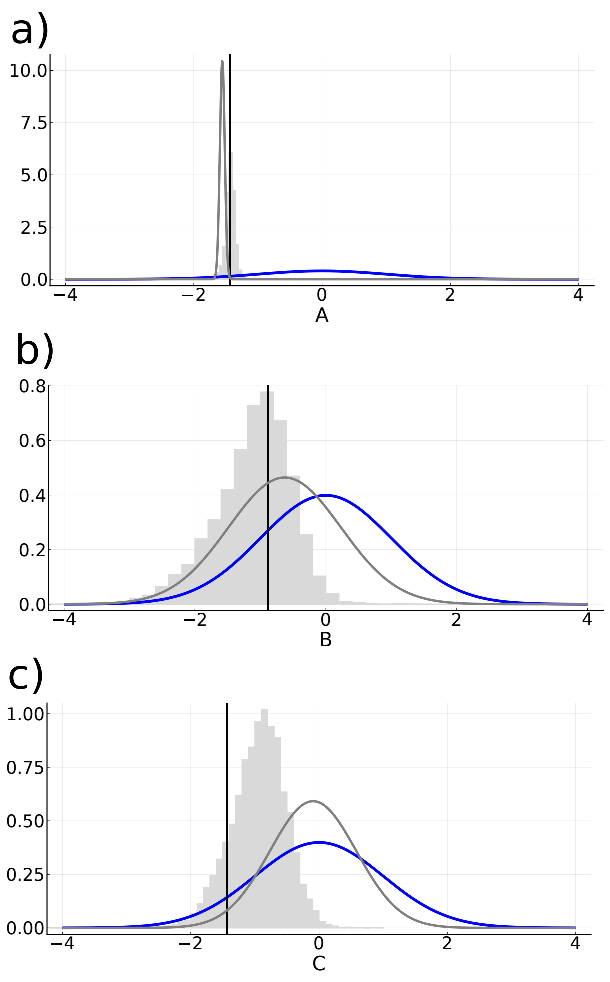

Figure 8. Histograms of MCMC samples of the posterior distributions obtained by the CES algorithm, for the parameters a, b, and c of a piecewise kernel (Eq. 14), given the zeroth, first, and second moment of the PMD at time $t_{\text{end}} = 60$ s. The prior distribution of the parameters is shown in blue, and the true parameter values are marked by vertical black lines. The Gaussian approximations to the posterior distributions shown as grey lines are specified bythe ensemble mean and standard deviation of the parameters in the last EKS iteration. The plot is shown in the unbounded parameter space.

lying parameters. Choosing prior parameter ranges that give rise to informative data while not being overly narrow required some exploration of the parameter space.

The problem of finding data that are informative enough to constrain microphysical processes is not limited to this proof-of-concept study, where the only data available are moments of the PMD. In fact, the space- and ground-based observations available today generally remain incomplete for directly constraining individual microphysical process rates in schemes (Morrison, van Lier-Walqui, Fridlind, et al., 2020), and choosing and combining data for use in Bayesian inversion will arguably be the ultimate challenge in developing data-informed microphysics schemes.

6 Summary and Conclusions

This paper introduces Cloudy, a flexible microphysics scheme that simulates collision-coalescence and collisional breakup of cloud droplets. We have shown how parameters of the collision kernels describing these droplet-droplet interactions can be learned from data through a computationally efficient Bayesian inversion.

The main points of this study can be summarized as follows:

- Cloudy is a bulk scheme for the collision-coalescence and collisional breakup of cloud droplets. By virtue of its flexible and modular design, the number of prognostic moments can be adjusted to the requirements of the particle mass distribution (PMD), and both PMD and collision kernel can easily be changed. Cloudy is broadly similar to BOSS, the scheme introduced by Morrison, van Lier-Walqui, Kumjian, and Prat (2020), with important differences in how the closure problem is formulated.
- We have looked at microphysics parameterizations through the lens of Bayesian inverse problems and have configured Cloudy to learn parameters of collision kernels from data using Calibrate-Emulate-Sample (CES; Cleary et al., 2021).
- CES is a three-stage approach to Bayesian inversion that is about a factor 1000 faster than traditional techniques. It makes estimation and uncertainty quantification of unknown parameters possible for computationally expensive and/or noisy models.
- CES is able to retrieve posterior parameter distributions in a suite of perfect-model experiments where Cloudy itself generates the data used to constrain the scheme. Results of experiments with different collision kernels show that most posterior distributions capture the true parameter values within 5% of the posterior mass.
- Moving beyond perfect-model experiments, we have learned collision kernel parameters from output generated by PySDM (Bartman et al., 2021), a Lagrangian particle-based microphysics model. In this experiment, we represent model error resulting from the closure assumption in Cloudy (an assumption that PySDM does not need to make) as a simple bias term. This modification in the setup of the inverse problem allows CES to retrieve the posterior distribution of the "true" parameter, not of that which minimizes the mismatch with the PySDM data.

Taken together, they constitute a proof of concept that informing microphysics schemes with data through Bayesian learning is possible in a computationally efficient way. This makes data-informed but physics-based approaches to modeling microphysics a more attractive and viable avenue for future parameterization development. Such approaches have been gaining traction in recent years (e.g., van Lier-Walqui et al., 2020; Morrison, van Lier-Walqui, Fridlind, et al., 2020) as they incorporate the existing physical understanding of microphysical processes while taking advantage of data and statistical tools to bridge knowledge gaps. Bayesian methods are particularly well suited to this task because they allow for continuous updates as the physical theory and understanding of these processes evolve. However, these strengths can only be brought to bear in combination with microphysics schemes that can be constrained in a rigorous and transparent way.

Ultimately, the goal is to inform these schemes by natural observations of clouds and precipitation. This will require methods that account for structural uncertainty, which is uncertainty resulting from the inadequacy of a model to reproduce a given set of data even with the "correct" parameter values, e.g., because it lacks some processes that have been present in producing the data. Neglecting to account for structural uncertainty results in parameter estimates that do not necessarily represent the true physics but that minimize the mismatch between the model output and the given data and hence maximize the predictive accuracy of the emulator (e.g., Kennedy & O'Hagan, 2001). Methods for quantifying structural uncertainty are less well developed than those for parametric uncertainty, but an established approach is to model the structural error as a Gaussian process at the interface of model and data (Kennedy & O'Hagan, 2000). An alternative is to use Gaussian processes or other machine learning techniques—for example, neural networks or learning from a dictionary of candidate terms (Brunton et al., 2016; Schneider et al., 2021)—directly where structural model errors actually occur, for example, in the collision kernel. In our example, the direct correspondence of the collision and breakup kernels between Cloudy and PySDM allowed us to instead use a simple additive bias term. However, incorporating structural uncertainty in a rigorous way will be a crucial element to fully exploit the potential of Bayesian inversion in constraining microphysics models.

Appendix A Monte Carlo Integration

Monte Carlo integration is a numerical technique that uses random numbers to approximate integrals. The core idea is to estimate the integral to be calculated by the sample mean of a sequence of random numbers, whose expected value is the exact value of the integral. There exist many variants and modifications of Monte Carlo integration that aim to reduce the variance of the estimator and hence the number of samples needed to achieve the desired accuracy.

Our goal here is to provide some implementation details of the Monte Carlo integration that is used to compute the time rate of change of the PMD moments in Cloudy, i.e., the right-hand side of Eq. 7. For a comprehensive treatment of Monte Carlo integration, the reader is referred to, e.g., Robert and Casella (2005).

Suppose that the integrand $h(x)$ (where $x \in \mathbb{R}^d$) can be written as a product of a function $T(x)$ and a probability density $p(x)$. We want to estimate the value of the integral

$$I = \int h(x)\ \mathrm{d}x = \int T(x)p(x)\ \mathrm{d}x \ = \ \mathbb{E}_p\left[T(x)\right] \tag{A1}$$

Monte Carlo integration consists of generating samples $\{X_1, \cdots X_N\}$ from the density p and approximating the integral (A1) by

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N} T(X_i) \tag{A2}$$

Due to the strong law of large numbers, $\hat{I}_N$ converges to I with probability 1 as $N \to \infty$. If the variance σ^2 of $T(X_i)$ is finite, then the standard error, $\sigma^2_{\bar{x}}$, is given by

$$\sigma^2_{\bar{x}} \coloneqq \mathbb{E}\left[(\hat{I}_N - I)^2\right] = \frac{\sigma}{N} \tag{A3}$$

The error is thus independent of the dimension d of the integral.

To compute the coalescence integral (see Eq. 7; repeated here for convenience),

$$I_{k,\mathrm{coal}} = \frac{1}{2}\int_0^\infty \int_0^\infty \left((m+m')^k - m^k - m'^k\right) C(m,m')f(m)f(m')\mathrm{d}m\,\mathrm{d}m', \tag{A4}$$

we define

$$p1(x,y) = \frac{1}{M_0^2} f(x)f(y). \tag{A5}$$

Drawing random samples $\{X_i\}_{i=1}^N$ and $\{Y_i\}_{i=1}^N$ from $p1$, $I_{k,\mathrm{coal}}$ can be estimated by

$$\hat{I}_{N,k,\text{coal}} = \frac{M_0^2}{2N}\sum_{i=1}^{N}((X_i+Y_i)^k - X_i^k - Y_i^k)C(X_i,Y_i) \tag{A6}$$

For the computation of the breakup integral $I_{k,\text{breakup}}$ the source and sink term are treated separately:

$$\begin{aligned} I_{k,\text{ breakup}} &= \frac{1}{2}\int_0^{m'+m''} m^k P(m;m',m'')\mathrm{d}m \int_0^\infty f(m')\ \mathrm{d}m' \int_0^\infty f(m'')B(m',m'')\ \mathrm{d}m'' \\ &- \int_0^\infty m^k f(m)\ \mathrm{d}m \int_0^\infty \frac{f(m'')B(m,m'')}{m+m''}\ \mathrm{d}m'' \int_0^{m+m''} m'P(m';m,m'')\ \mathrm{d}m' \\ &=: I_{k,\text{ breakup}}^{\text{source}} - I_{k,\text{ breakup}}^{\text{sink}} \end{aligned} \tag{A7}$$

Both terms ($I_{k,\text{ breakup}}^{\text{source}}$ and $I_{k,\text{ breakup}}^{\text{sink}}$) allow for the analytical integration of one of the variables, such that the Monte Carlo integration is only applied to the remaining double integrals. To achieve this, we make use of the fact that parts of the fragment distribution, $P(m;m',m'') = \beta^2(m'+m'')\exp(-\beta m)$ (Eq. 3), can be combined with m^k (for $I_{k,\text{ breakup}}^{\text{source}}$) and m' (for $I_{k,\text{ breakup}}^{\text{sink}}$) into Gamma distributions with known cumulative density functions (cdf). We start with the breakup source integral, where introducing a normalization factor $\gamma = \frac{\beta^{\alpha_k-2}}{\Gamma(\alpha_k)}$ allows for the construction of Gamma(α_k,β) distributions with density

$$p_{2,k}(m) = \frac{\beta^{\alpha_k}}{\Gamma(\alpha_k)} m^{\alpha_k-1}\exp(-\beta m), \tag{A8}$$

with $\alpha_k = k{+}1$. Outside of $p_{2,k}(m)$, there is no dependence on m left in $I_{k,\text{ breakup}}^{\text{source}}$, and so the integral $\int_0^{m'+m''} p_{2,k}(m)\frac{1}{\gamma}\mathrm{d}m = \frac{1}{\gamma}$ can be integrated separately. Its solution is the cdf of a Gamma(α_k,β) distribution, which for positive integers α_k is given by

$$F(x;\alpha_k,\beta) = 1 - \sum_{i=0}^{\alpha_k-1}\frac{(\beta x)^i}{i!}\exp(-\beta x), \tag{A9}$$

evaluated at $x = m'+m''$. The remaining double integral is then computed using the same technique as for the coalescence integral: Drawing random samples $\{Y_i\}_{i=1}^N$ and $\{Z_i\}_{i=1}^N$ from $p1(x,y) = \frac{1}{M_0^2}f(x)f(y)$, the Monte Carlo estimate of the breakup source term is

$$\hat{I}_{N,k,\text{ breakup}}^{\text{source}} = \frac{M_0^2}{2\gamma N}\sum_{i=1}^{N}(Y_i+Z_i)B(Y_i,Z_i)F(Y_i+Z_i;\alpha_k,\beta) \tag{A10}$$

A similar approach is applied to the breakup sink integral, which is simplified to a double integral by the analytic integration of the density of a Gamma$(2,\beta)$ distribution (resulting from the product of m' and $\beta^2\exp(-\beta m)$. Drawing random samples $\{X_i\}_{i=1}^N$ and $\{Z_i\}_{i=1}^N$ from $p1$, the breakup sink term is approximated by

$$\hat{I}_{N,k,\text{ breakup}}^{\text{sink}} = \frac{M_0^2}{N}\sum_{i=1}^{N}X_i^k B(X_i,Z_i)F(X_i+Z_i;2,\beta). \tag{A11}$$

Putting everything together, the time rate of change of the k-th PMD moment due to collision-coalescence and collisional breakup (Eq. 7) is approximated by

$$\begin{aligned}\frac{\partial M_k}{\partial t} &= I_{k,\text{coal}} + I_{k,\text{breakup}} \\ &\approx \hat{I}_{N,k,\text{coal}} + \hat{I}_{N,k,\text{ breakup}} \\ &= \hat{I}_{N,k,\text{coal}} + \hat{I}^{\text{source}}_{N,k,\text{ breakup}} - \hat{I}^{\text{sink}}_{N,k,\text{ breakup}}.\end{aligned} \tag{A12}$$

The results shown in this paper were obtained using $N = 200$ Monte Carlo samples.

Appendix B Superdroplet Breakup

The superdroplet method to model stochastic collision-coalescence of droplets is extended in this work to include a breakup-like process for numerical experiments. As breakup introduces a competing process for coalescence, this stochastic breakup implementation provides a more realistic set of dynamics to demonstrate the CES algorithm's ability to learn from data. Here we will briefly describe the modifications made to the existing package PySDM (Bartman & Arabas, 2021) in order to introduce a breakup-like process.

Maintaining the notation of (Shima et al., 2009), each superdroplet with label index i for these box model simulations is assigned a multiplicity ξ_i and a mass m_i. A pair α of superdroplets collides and coalesces with scaled probability p_α, and a random number ϕ is generated to determine the number of coalescences that occur, γ_α. In the implementation of PySDM, p_α is computed based on a collision rate from kinetic theory and a coalescence efficiency E_c, which combined are referred to as the coalescence kernel. In this new implementation, every time step also includes the potential for collisional breakup of a given superdroplet pair. Like p_α, the probability of a breakup is computed based on the collision rate multiplied with $(1 - E_c)$, and whether the breakup occurs is determined based on the generation of a new random number.

For a collision-coalescence event that occurs for the SD pair α with γ_α coalescences, the multiplicities of the superdroplets are updated as follows:

$$\begin{aligned}\xi_j &\leftarrow \xi_j - \gamma_\alpha \xi_k \\ \xi_k &\leftarrow \xi_k \\ M_j &\leftarrow M_j \\ M_k &\leftarrow M_k + \gamma_\alpha M_j\end{aligned} \tag{B1}$$

In this process, the superdroplet j maintains its mass but loses multiplicity to coalescence, while droplets k grow due to coalescence with droplets j.

As a substitute for collisional breakup, we treat the process as a collisional coalescence of two superdroplets followed by spontaneous breakup into n_f uniform fragments. Thus if a breakup is determined to occur, the same quantity γ_α is computed to determine the number of pre-coalescences that occur according to the same dynamics described above. Subsequently, the newly coalesced superdroplet k spontaneously fragments:

$$\begin{aligned}\xi_k &\leftarrow n_f \xi_k \\ M_k &\leftarrow M_k / n_f\end{aligned} \tag{B2}$$

For the purposes of these simple test cases, n_f is set to 2 such that the resulting SD has the average mass of the two colliding SDs. However, we note that this choice is fundamentally unrealistic, as it would drive the system toward uniformly sized droplets that are the average size of the initial distribution. For the numerical experiments presented in this paper, the simulation time is chosen to be far shorter than the time required for this nonstochastic behavior to become apparent.

Appendix C Open Research

The code for the Calibrate-Emulate-Sample algorithm is available at `https://github.com/CliMA/CalibrateEmulateSample.jl`. The Cloudy repository can be found at `https://github.com/CliMA/Cloudy.jl`.

Acknowledgments
This work was supported by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program, by the Hopewell Fund, and the Paul G. Allen Family Foundation.

References

Andrejczuk, M., Grabowski, W. W., Reisner, J., & Gadian, A. (2010). Cloud-aerosol interactions for boundary layer stratocumulus in the Lagrangian cloud model. *J. Geophys. Res. Atmos.*, *115*(D22). doi: 10.1029/2010JD014248

Bartman, P., & Arabas, S. (2021). *Super-droplet kernels – backend-level routines for Monte-Carlo particle coagulation solvers: API proposal with CPU and GPU implementation in Python.*

Bartman, P., Arabas, S., Górski, K., Jaruga, A., Łazarski, G., Olesik, M., … Talar, A. (2021). *PySDM v1: particle-based cloud modelling package for warm-rain microphysics and aqueous chemistry.*

Beard, K. V., & Ochs, H. T. (1995). Collisions between small precipitation drops. Part II: Formulas for coalescence, temporary coalescence, and satellites. *J. Atmos. Sci.*, *52*(22), 3977–3996. doi: 10.1175/1520-0469(1995)052⟨3977:CBSPDP⟩2.0.CO;2

Beheng, K. D. (2010). The evolution of raindrop spectra: A review of microphysical essentials. In *Rainfall: State of the science* (pp. 29–48). American Geophysical Union (AGU). doi: 10.1029/2010GM000957

Best, D. J. (1983). A note on Gamma variate generators with shape parameter less than unity. *Computing*, *30*(2), 185–188. doi: 10.1007/BF02280789

Borowka, S., Heinrich, G., Jahn, S., Jones, S., Kerner, M., & Schlenk, J. (2019). A GPU compatible quasi-Monte Carlo integrator interfaced to pySecDec. *Comp. Phys. Comm.*, *240*, 120–137. doi: 10.1016/j.cpc.2019.02.015

Brockwell, P. J., & Davis, R. A. (1996). *Introduction to time series and forecasting.* Springer New York.

Brunton, S., Proctor, J., & Kutz, J. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci.*, *113*, 3932–3937.

Cleary, E., Garbuno-Inigo, A., Lan, S., Schneider, T., & Stuart, A. M. (2021). Calibrate, emulate, sample. *J. Comp. Phys.*, *424*. doi: 10.1016/j.jcp.2020.109716

Dunbar, O. R. A., Garbuno-Inigo, A., Schneider, T., & Stuart, A. M. (2021). Calibration and uncertainty quantification of convective parameters in an idealized GCM. *J. Adv. Model. Earth Syst.*, *13*(9), e2020MS002454. doi: https://doi.org/10.1029/2020MS002454

Duncan, A. B., Stuart, A. M., & Wolfram, M.-T. (2021). *Ensemble inference methods for models with noisy and expensive likelihoods.*

Dziekan, P., & Pawlowska, H. (2017). Stochastic coalescence in Lagrangian cloud microphysics. *Atmos. Chem. Phys.*, *17*(22), 13509–13520. doi: 10.5194/acp-17-13509-2017,2017

Feingold, G., Tzivion, S., & Leviv, Z. (1988). Evolution of raindrop spectra. Part I: Solution to the stochastic collection/breakup equation using the method of moments. *J. Atmos. Sci.*, *45*(22), 3387–3399. doi: 10.1175/1520-0469(1988)045⟨3387:EORSPI⟩2.0.CO;2

Garbuno-Inigo, A., Hoffmann, F., Li, W., & Stuart, A. M. (2020). Interacting Langevin diffusions: Gradient structure and ensemble Kalman sampler. *SIAM J. Appl. Dyn. Syst.*, *19*(1), 412–441. doi: 10.1137/19M1251655

Garbuno-Inigo, A., Nüsken, N., & Reich, S. (2020). Affine invariant interacting Langevin dynamics for Bayesian inference. *SIAM J. Appl. Dyn. Syst.*, *19*(3), 1633–1658. doi: 10.1137/19M1304891

Golovin, A. M. (1963). On the kinetic equation for coagulating cloud droplets with allowance for condensation. *Izv. Acad. Sci. USSR, Ser. Geophys*, *10*, 949–953.

Howland, M. F., Dunbar, O., & Schneider, T. (2021). Parameter uncertainty quantification in an idealized GCM with a seasonal cycle. *ESSOAr*, 26.

Iglesias, M. A., Law, K. J. H., & Stuart, A. M. (2013). Ensemble Kalman methods for inverse problems. *Inverse Probl.*, *29*(4), 045001. doi: 10.1088/0266-5611/29/4/045001

Kanzaki, J. (2011). Monte Carlo integration on GPU. *Eur. Phys. J. C*, *71*(2).

Kennedy, M. C., & O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, *87*(1), 1–13.

Kennedy, M. C., & O'Hagan, A. (2001). Bayesian calibration of computer models. *J. R. Stat. Soc.: Series B (Statistical Methodology)*, *63*(3), 425–464. doi: 10.1111/1467-9868.00294

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Kleijnen, A. A. N., Jack P. C.and Ridder, & Rubinstein, R. Y. (2013). Variance reduction techniques in Monte Carlo methods. In S. I. Gass & M. C. Fu (Eds.), *Encyclopedia of operations research and management science* (pp. 1598–1610). Boston, MA: Springer. doi: 10.1007/978-1-4419-1153-7_638

Liu, C., Martin, R., & Syring, N. (2017). Efficient simulation from a gamma distribution with small shape parameter. *Comp. Stat.*, *32*(4), 1767–1775. doi: 10.1007/s00180-016-0692-0

Long, A. B. (1974). Solutions to the droplet collection equation for polynomial kernels. *J. Atmos. Sci.*, *31*(4), 1040–1052. doi: 10.1175/1520-0469(1974)031⟨1040:STTDCE⟩2.0.CO;2

Low, T. B., & List, R. (1982). Collision, coalescence and breakup of raindrops. Part I: Experimentally established coalescence efficiencies and fragment size distributions in breakup. *J. Atmos. Sci.*, *39*(7), 1591–1606. doi: 10.1175/1520-0469(1982)039⟨1591:CCABOR⟩2.0.CO;2

Morrison, H., Kumjian, M. R., Martinkus, C. P., Prat, O. P., & van Lier-Walqui, M. (2019). A general n-moment normalization method for deriving raindrop size distribution scaling relationships. *J. Appl. Meteorol. Climatol.*, *58*(2), 247–267. doi: 10.1175/JAMC-D-18-0060.1

Morrison, H., van Lier-Walqui, M., Fridlind, A. M., Grabowski, W. W., Harrington, J. Y., Hoose, C., … Xue, L. (2020). Confronting the challenge of modeling cloud and precipitation microphysics. *J. Adv. Model. Earth Syst.*, *n/a*(n/a), e2019MS001689. doi: 10.1029/2019MS001689

Morrison, H., van Lier-Walqui, M., Kumjian, M. R., & Prat, O. P. (2020). A Bayesian approach for statistical-physical bulk parameterization of rain microphysics. Part I: Scheme description. *J. Atmos. Sci.*, *77*(3), 1019–1041. doi: 10.1175/JAS-D-19-0070.1

Posselt, D. J. (2016). A Bayesian Examination of Deep Convective Squall-Line Sensitivity to Changes in Cloud Microphysical Parameters. *J. Atmos. Sci.*, *73*(2), 637–665. doi: 10.1175/JAS-D-15-0159.1

Posselt, D. J., & Vukicevic, T. (2010). Robust characterization of model physics uncertainty for simulations of deep moist convection. *Mon. Wea. Rev.*, *138*(5), 1513–1535. doi: 10.1175/2009MWR3094.1

Pruppacher, H., & Klett, J. (1978). *Microphysics of clouds and precipitation* (1st ed.). Reidel.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Cambridge, MA, USA: MIT Press.

Riechelmann, T., Noh, Y., & Raasch, S. (2012). A new method for large-eddy simulations of clouds with Lagrangian droplets including the effects of turbulent collision. *New J. Phys.*, *14*(6), 065008. doi: 10.1088/1367-2630/14/6/065008

Robert, C. P., & Casella, G. (2005). *Monte Carlo statistical methods*. Berlin, Heidelberg: Springer Verlag.

Schneider, T., Lan, S., Stuart, A., & Teixeira, J. (2017). Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations. *Geophys. Res. Lett.*, *44*(24), 12,396–12,417. doi: 10.1002/2017GL076101

Schneider, T., Stuart, A. M., & Wu, J. (2021). Ensemble Kalman inversion for sparse learning of dynamical systems from time-averaged data. *J. Comp. Phys.*.

Shima, S., Kusano, K., Kawano, A., Sugiyama, T., & Kawahara, S. (2009). The super-droplet method for the numerical simulation of clouds and precipitation: a particle-based and probabilistic microphysics model coupled with a non-hydrostatic model. *Q. J. R. Meteorol. Soc.*, *135*(642), 1307–1320. doi: 10.1002/qj.441

Shima, S., Sato, Y., Hashimoto, A., & Misumi, R. (2019). Predicting the morphology of ice particles in deep convection using the super-droplet method: development and evaluation of SCALE-SDM 0.2.5-2.2.0/2.2.1. *Geosci. Model Dev. Discuss.*, *2019*, 1–83. doi: 10.5194/gmd-13-4107-2020

Smoluchowski, M. V. (1916). Drei Vorträge über Diffusion, Brownsche Bewegung und Koagulation von Kolloidteilchen. *Z. Phys.*, *17*, 557–585.

van Lier-Walqui, M., Morrison, H., Kumjian, M. R., Reimel, K. J., Prat, O. P., Lunderman, S., & Morzfeld, M. (2020). A Bayesian approach for statistical–physical bulk parameterization of rain microphysics. Part II: Idealized Markov chain Monte Carlo experiments. *J. Atmos. Sci.*, *77*(3), 1043–1064. doi: 10.1175/JAS-D-19-0071.1

van Lier-Walqui, M., Vukicevic, T., & Posselt, D. J. (2014). Linearization of microphysical parameterization uncertainty using multiplicative process perturbation parameters. *Mon. Wea. Rev.*, *142*(1), 401–413. doi: 10.1175/MWR-D-13-00076.1

Weisberg, S. (2014). *Applied linear regression* (Fourth ed.). Hoboken NJ: Wiley.