Probabilistic and data-driven methods for numerical PDEs

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Johannes Krotz

June 2024

To Dominique & Lisa, whose wedding I missed to defend this dissertation.
All the luck and love in the world for your future together ♡

And to all my friends, family, and everyone else who did not make it past the title: Someone tell them, or they might never know they are mentioned here.

Lastly, to everyone who is along for the ride: Take Figure 1 as a warning.

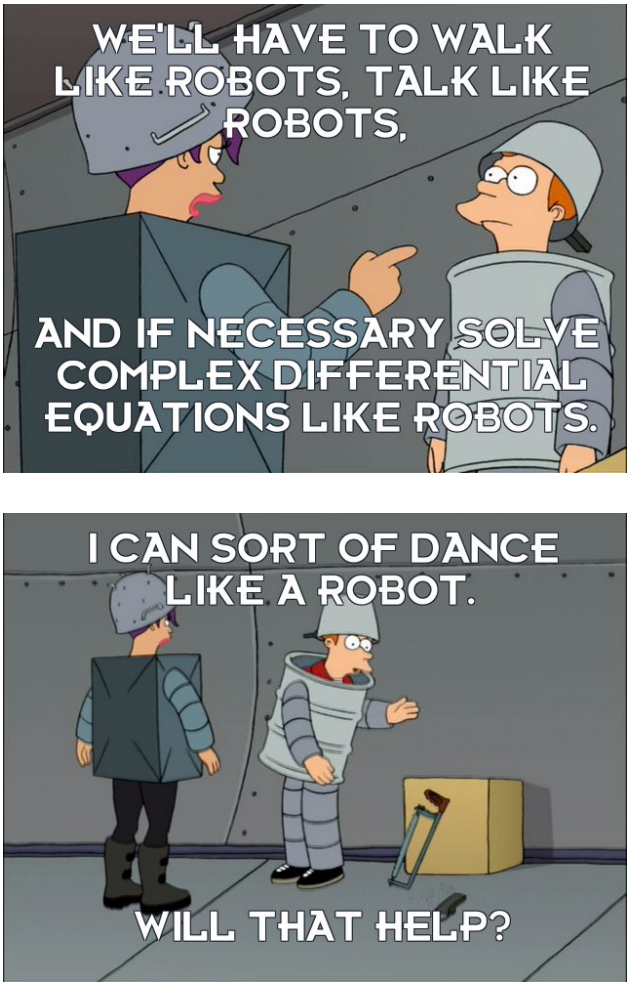


Figure 1: Scene from Futurama Season 1 Episode 5: "Fear of a Bot Planet". Images generated through morbotron.com.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Juan Restrepo, who was not only a good friend but also provided me with a plethora of advice and opportunities throughout this journey. I am indebted to my collaborators Carl Gable, Cory Hauck, Jeff Hyman, Jorge Ramires, Matt Sweeney, and Ryan McClarren. Their perspectives and insights have enriched my understanding, broadened my horizons, and enabled me to complete the projects presented in this dissertation.

I am also thankful to the remaining members of my committee, Suzanne Lenhart and Vasileios Maroulas.

My heartfelt appreciation goes to my family, friends, and colleagues for their encouragement, support, and stimulating discussions, as well as for grounding and distracting me. I blame some of you for not being done sooner. Thanks!

Finally, I would like to thank the NSF, who funded several of the projects below through the NSF MSGI.

Abstract

This dissertation consists of three integral self-contained parts. The first part develops a novel Monte Carlo algorithm, called the near-Maximal Algorithm for Poisson-disk Sampling (nMAPS), to efficiently generate the nodes of a high-quality mesh for the calculation of flow and the associated transport of chemical species in low-permeability fractured rock, such as shale and granite. A good mesh balances accuracy requirements with a reasonable computational cost, i.e., it is generated efficiently, dense where necessary for accuracy, and contains no cells that cause instabilities or blown-up errors. Quality bounds for meshes generated through nMAPS are proven, and its efficiency is demonstrated through numerical experiments.

In the second part, a deterministic Monte Carlo hybrid method for time-dependent problems based on the physics of particle transport described through the linear Boltzmann equation is presented. The method splits the system into collided and uncollided particles and treats these sets with different methods. Uncollided particles are handled through high-accuracy Monte Carlo methods, while the density of collided particles is calculated using discontinuous Galerkin methods. Theoretical details of the algorithm are developed and shown to be effective through numerical experiments. The properties associated with the labeling as collided and uncollided leverage the respective strengths of these methods, allowing for overall more accurate and computationally efficient solving than each method on its own.

In the last chapter, an extension to the Dynamic Likelihood Filter (DLF) is presented to include Advection-Diffusion equations. The DLF is a Bayesian estimation method specifically designed for wave-related problems. It improves on traditional methods, such as variants of Kalman filters, by not only using data at its time of observation but also at later times

by propagating observations forward through time. This enriches the available data and improves predictions and uncertainties. The theory to include diffusion in the framework of the DLF is developed, and it is shown through numerical experiments that the DLF outperforms traditional data assimilation techniques, especially when observations are precise but sparse in space and time.

Table of Contents

List of Figures

Chapter 1

Introduction and Outline

Partial Differential Equations (PDEs), that is equations which depend on various partial derivatives of their solution functions, have, for a long time, been a cornerstone in natural sciences such as physics and engineering. They are fundamental in our understanding of the world and are at the core of some of the most influential theories of our time such as Quantum theory, Fluid dynamics, General relativity and many more. Outside of the natural Sciences their range of applications stretches from purely Mathematical considerations like Differential Geometry or Variational Calculus to the description of complex systems in Computer Science, Neuroscience and Social Sciences. Stochastic Partial Differential Equations (SPDEs), that generalize PDEs through the inclusion of random variables as Forces and coefficients, are widely applied in Financial Sciences, Statistical Mechanics, Weather and Climate modeling and many more areas. While there is a vast and still growing body of theories discussing and classifying the existence and uniqueness and various other properties of solutions to PDEs, it is widely understood that finding an explicit solution to a given equation by hand is the exception not the rule, particularly as systems become more complex. In applications approximate solutions are usually generated through numerical methods using computers.

Following the 'No free lunch theorem' acceptable accuracy of such an approximation usually needs to be balanced with the computational complexity required to find it. While this has led to the development of better and bigger hardware, the fact that accuracy of solvers and complexity often times do not relate linearly makes simple upscaling of a problem prohibitively expensive in many cases. Thus the development of methods with a better

trade-off between accuracy and computational complexity is a very active area of research tackled by experts in many fields. Such algorithmic attempts at tackling this issue include, but are not limited to:

- tailoring the way of approximation to a specific problem.
- direct simulation of systems using deterministic or Monte Carlo methods.
- the application of Machine learning methods such as Deep neural networks either instead or as part of an existing solver.
- the incorporation of real-life data to improve the predictions of solvers.

The following chapters of this dissertation are comprised of three manuscripts that tackle the issue of improving accuracy of solutions to PDEs or the predictions derived from them via one or more of the ideas just listed. They are structured as follows.

The manuscript in Chapter two, 'Variable resolution Poisson-disk sampling for meshing discrete fracture networks' [3], covers an approach for the transport equation that tailors the discretization of the problem to the specific problem. The main body of work focuses on an algorithm that can provide this discretization quicker than previously existing methods, thus leading to a net speedup in computation time. The algorithm is specifically designed to provide meshes for Discrete Fracture Networks (DFN), which are commonly used in the simulation of fractured materials. DFN explicitly simulates fractures with the same distribution as the fractures in the actual material. Our two-phase algorithm first randomly generates a 2D mesh on the fractures. It then utilizes this 2D mesh as a foundation to construct a 3D mesh on the surrounding material. The nodes of both the initial 2D and the 3D mesh are generated through Poisson disk sampling. Poisson disk samples exhibit provably favorable properties for 2D triangulation. We establish stringent bounds for mesh quality in the 2D part of the algorithm and numerically test their validity. In 3D, Poisson disk samples do not inherently ensure high-quality meshes. However, we found that in 3D triangulations of Poisson disk samples, low-quality elements of a mesh are exceedingly rare. Through numerical experimentation with the 3D part of our algorithm, we demonstrate that

rejecting nodes that lead to low-quality elements and subsequently resampling is a successful strategy for obtaining high-quality 3D meshes within reasonable computation time.

Chapter three is based on the manuscript 'A Hybrid Monte Carlo, Discontinuous Galerkin method for linear kinetic transport equations' [1]. In it, we present an algorithm to solve the linear Boltzmann equation, a linear kinetic equation describing the free transport of particles scattering off a medium. It is used to model many systems, including neutronic dynamics, radiation transfer, cometary flow, and dust particles. Generalizations find applications in colloidal systems, fluid dynamics, and non-equilibrium thermodynamics. Common approaches to finding a solution are Monte Carlo methods using ideas based on direct simulation of the particles or discretization methods. Examples of the latter would be the use of Discontinuous Galerkin (DG) methods. While the latter reduces the problem to systems of linear equations, the coupling of the PDEs causes the dimension of these linear systems to become prohibitive at high accuracies. While Monte Carlo methods do not suffer from this curse of dimensionality, they become substantially less efficient if the number of scattering events is increased. Conveniently, DG methods excel in the diffusion limit, that is at intermediate to high scattering probabilities, producing acceptable results at relatively low cost. Thus MC and DG appear to complement each other well. We put this fact to the test in this manuscript.

The algorithm presented in chapter three combines MC and DG into a hybrid method that splits the original PDE into scatter-free subequations that are perfectly suited for high accuracy Monte Carlo methods and another set of subequations that are taking care of the scattering. Through smart handling of the interactions of these split equations, we see that it suffices to solve the scattering equations with relatively cheap, low accuracy DG-methods without a substantial loss in overall accuracy. The result is a method that is comparable in accuracy and run time, even though not strictly better than MC in low scattering cases and significantly more efficient than MC and DG respectively in cases with higher scattering without sacrificing accuracy.

Lastly, in chapter four, titled 'A Dynamic Likelihood Approach to Filtering for Advection Diffusion Processes' [2], a Bayesian filtering algorithm is introduced. Such filtering algorithms use a noisy/inaccurate model along with also noisy data to describe a real-life system. In this case, the real system is assumed to be governed by a stochastic PDE. The goal is to find

the best estimator of the true state of a system conditioned on the data and the model. This adds a new dimension to the notion of accuracy, as we are now not only interested in how well a numerical solution approximates an analytical solution of a PDE. Now we care about how well a real system is approximated, adding a modeling error on top of the numerical errors considered so far. Classical algorithms will minimize the expected error between Model and reality at any time at which data is available. The dynamic likelihood filter introduced in this chapter extends this by using the available data to generate simulated data at times between observations. While this adds an additional modeling error we can show that minimizing the expected error not only through real but also simulated data yields a net increase in accuracy in prediction in settings where data is sparse, but accurate. Specifically we show how this method can be used in the context of stochastic advection diffusion equation and compare the newly developed method to a Kalman filter in the same context. We demonstrate numerically that our method, the Dynamic likelihood filter (DLF), outperforms the classic filter in terms of accuracy in many cases. We show advantages specifically in settings with sparse, but accurate data as well as very inaccurate or even ill-informed models. An interesting feature of the DLF is its capability of providing Bayesian estimates at future times.

Lastly, in Chapter Four, titled 'A Dynamic Likelihood Approach to Filtering for Advection Diffusion Processes' [2], an introduction is made to a Bayesian filtering algorithm. Such filtering algorithms use a noisy/inaccurate model along with noisy data to describe a real-life system. This introduces an additional aspect to the concept of accuracy. Now, we not only seek to understand how well a numerical solution approximates an analytical solution of a PDE. Classical algorithms will minimize the expected error between model and reality at any time at which data is available, utilizing real observations only. However, our approach extends this by incorporating simulated data, which we refer to as pseudo-Observations. We demonstrate how this can be done specifically for systems governed by stochastic advection diffusion equations. In cases where advection dominates information is travels along the characteristics of the PDE, which is reflected in the fact that we evolve pseudo-observations along these characteristics. Accounting for difusion and noise along these characteristics is non-trivial and requires some feedback from the model. Although having to simulate pseudo-observations introduces an additional modeling error, we can demonstrate that minimizing the expected

error through both real and simulated data leads to a net increase in prediction accuracy, particularly in settings where data is sparse but accurate. We demonstrate numerically that our method, the Dynamic likelihood filter (DLF), outperforms the classical Kalman filter in terms of accuracy in many cases. We show advantages specifically in settings with sparse, but accurate data as well as very inaccurate or even ill-informed models. An interesting feature of the DLF is its capability of providing Bayesian estimates at future times.

References

[1] Johannes Krotz, Cory D. Hauck, and Ryan G. McClarren. A hybrid monte carlo, discontinuous galerkin method for linear kinetic transport equations, 2023. 3

[2] Johannes Krotz, Juan M. Restrepo, and Jorge M. Ramires. A likelihood approach to filtering for advection diffusion processes. Manuscript in preparation, 2024. 3, 4

[3] Johannes Krotz, Matthew R. Sweeney, Carl W. Gable, Jeffrey D. Hyman, and Juan M. Restrepo. Variable resolution poisson-disk sampling for meshing discrete fracture networks. *Journal of Computational and Applied Mathematics*, 407:114094, 2022. 2

Chapter 2

Variable resolution Poisson-disk sampling for meshing discrete fracture networks

2.1 Disclosure

This chapter is, up to formatting identical to the paper of the same name [40]. The paper is a collaborative work with Matthew R. Sweeney, Carl W. Gable Jeffrey D. Hyman and Juan M. Restrepo and was published in the Journal of Computational and Applied Mathematics in 2022. Ideas and theoretical results newly presented in this paper were worked out by Johannes Krotz under their guidance. Numerical experiments presented were executed by Johannes Krotz using software implemented in Python also by Johannes Krotz. All coauthors wrote and edited the paper together.

2.2 Abstract

We propose a two-stage algorithm for generating Delaunay triangulations in 2D and Delaunay tetrahedra in 3D that employs near maximal Poisson-disk sampling. The method generates a variable resolution mesh in linear run time. The effectiveness of the algorithm is demonstrated by generating an unstructured 3D mesh on a discrete fracture network (DFN). 2D Poisson-disk samplings on the DFN are generated through a cell-based rejection algorithm. After an initial sampling the grid-cells are used to fill in holes in the sampling in order to obtain a near-maximal Poisson-disk sampling. The 2D-sample on the DFN is then used as seed for a 3D algorithm, that generates a conforming 3D-Poisson-disk sampling on the surrounding volume of the DFN. Low quality tetrahedra are removed from the 3D-sampling and replaced in a resampling process. Even though Poisson-disk sampling methods do not provide triangulation quality bounds in more than two-dimensions, we found that low quality tetrahedra are infrequent enough and could be successfully removed to obtain high quality balanced 3-dimensional meshes with tetrahedra topologically acceptable for the application in DFN.

2.3 Introduction

There are a number of methods used to model flow and the associated transport of chemical species in low-permeability fractured rock, such as shale and granite. The most common are continuum models, which use effective medium parameters [21, 42, 56, 57, 68, 70] and

discrete fracture network/matrix (DFN) models, where fractures and the networks they form are explicitly represented [10, 44, 58]. In the DFN methodology, individual fractures are represented as planar $N-1$ dimensional objects embedded within an N dimensional space. Both conforming methods, where the mesh conform to intersections [30, 52, 53], and non-conforming methods, which use more complex discretization schemes so the mesh does not need to be conforming [5, 18, 59, 60], are currently in use. If the matrix surrounding the fracture network needs to be meshed, complications of mesh generation are compounded for both conforming and non-conforming methods [4]. While the explicit representation of fractures allows for DFN models to represent a wider range of transport phenomena and makes them a preferred choice when linking network attributes to flow properties [25, 31, 29], it also leads to unique and complex issues associated with mesh generation.

We propose a two stage algorithm that generates a conforming variable resolution triangular mesh on a three-dimensional discrete fracture network. The proposed algorithm uses maximal Poisson-disk sampling to efficiently produce the point distribution used to generate the mesh of each fracture with controlled mesh resolution. The first stage is based on the framework presented in [14], that uses a rejection algorithm to generate an initial Poisson-disk sampling with linear runtime in the number of points generated. The second phase is based on the framework presented in [49] and adds additional points to the samples. This second steps maximizes density without violating the restrictions of a Poisson-disk sampling.

Once a Poisson-disk sampling is generated, a conforming Delaunay algorithm [51] is used to connect this point distribution where lines of intersection between fractures form a set of connected edges in the Delaunay triangulation of the network. The time it takes to generate the samplings scales linearly with the number of nodes. While it is not guaranteed that the density of our Poisson-disk sampling is maximal, i.e. no further nodes can be added without violating the restrictions on distances between nodes, we show that in practice our samples are maximal enough to obtain high quality meshes. We also present a three-dimensional version of the method that can be used to create a tetrahdron mesh of the volume surrounding the network that conforms to the fracture network.

In section 2.4, we describe the challenges in the DFN mesh generation and the general properties of maximal Poisson-disk sampling. In section 2.5, we provide a detailed explanation

of our method, for both 2D fracture networks and 3D volume meshing. In section 2.6, we propose metrics to access the quality of the mesh and run times for both 2D and 3D demonstration examples. In section 2.7, we provide a few remarks.

2.4 Background

2.4.1 Discrete Fracture Networks: Mesh Generation Background

Due to the epistemic uncertainty associated with hydraulic and structural properties of subsurface fractured media, fracture network models are typically modeled probabilistically [54, 55, 57]. In the DFN methodology, individual fractures are placed into the computational domain with locations, sizes, and orientations that are sampled from appropriate distributions based on field site characterizations. The fractures form an interconnected network embedded within the porous medium. Each fracture must be meshed for computation, so that the governing equations for flow and transport can be numerically integrated to simulate physical phenomena of interest.

Formally, each fracture in a DFN can be represented as a planar straight-line graph (PSLG) composed of a set of line segments that represent the boundary of the fracture and a set of line segments that represent where other fractures intersect it. Then each fracture can be described by a set of boundary points on the PSLG, denoted $\{p\}$, and a set of intersection lines $\{\ell_{i,j}\}$, where the subscripts i and j indicate that this line corresponds to the intersection between the ith and jth fractures. Once $\{p\}$ and $\{\ell_{i,j}\}$ are obtained for every fracture in the network, a point distribution covering each fracture must be generated. If a conforming numerical scheme is used, then all cells of $\{\ell_{i,j}\}$ are discretized lines in the mesh which must coincide between intersecting fractures. So long as minimum feature size constrains are met, a conforming triangulation method, such as presented in [51], can be implemented to connect the vertices such that all lines of intersection form a set of connected edges in a triangulation.

In general, one wants to properly resolve all relevant flow and transport properties of interest while minimizing the number of nodes in the mesh, and these two goals compete. A uniform mesh resolution is straightforward to generate and appropriate for Eulerian transport

simulations. However, spatially variable numerical diffusion means that the resulting mesh will need a very small resolution to accurately capture solute fronts.[2] Variable mesh resolution can be appropriate for single-phase flow simulations or in particle tracking simulations where the spatially variable resolution does not adversely affect transport properties. However, this variable mesh generation is more complex than its uniform counterpart. One of the principal complications of variable mesh resolution generation is creating a smooth transition of cell sizes. Absent that, jumps in the computed fields of interest and other numerical artifacts will occur. the starting point for the notion of mesh quality would appear to be the analysis leading to the minimum angle condition that the smallest angle should be bounded away from zero. This originated with Zlamal [71] and is quoted by Strang and Fix [65] together with a statement regarding how poorly shaped triangles may have an effect on the condition number of the linear algebra problem that must be solved. This result was improved by Babuska and Aziz [1]. Most methods for the generation of a conforming DFN mesh use a uniform point distribution on the networks and then modify the connectivity locally to conform to intersections [52, 53]. When using a conforming mesh, the numerical methods for resolving flow and transport in the network are typically simpler and have fewer degrees of freedom compared to non-conforming mesh methods [20]. Similarly, almost all non-conforming numerical methods use a uniform resolution, but some create variable resolutions across fractures (still uniform within a single plane) in an attempt to reduce the number of total nodes in the mesh [6]. A variable mesh resolution in non-conforming schemes could drastically reduce the number of nodes in the mesh while retaining the the ability to retain higher orders of accuracy. However it is rarely implemented due to the associated meshing complications [8].

The generation of a variable resolution, unstructured conforming mesh is quite rare, even with the advantages noted above. One technique in use is the Features Rejection Algorithm for Meshing (FRAM) that addressed the issues associated with conforming DFN mesh creation by coupling it with network generation [30]. Through this technique, FRAM allows for the creation of a variable resolution mesh that smoothly coarsens away from intersections where pressure gradients are typically the highest in flow simulations. FRAM has been implemented in the computational suite dfnWorks [33], which has been used to probe fundamental

aspects of geophysical flows and transport in fractured media [27, 28, 35, 37, 46, 64] as well as practical applications including hydraulic fracturing operations [26, 38, 45], inversion of micro-seismicity data for characterization of fracture properties [50], the long term storage of spent civilian nuclear fuel [25], and geo-sequestration of carbon dioxide into depleted reservoirs [32].

However, the implementation used is an iterative refinement method for point distribution, which is very inefficient. To triangulate each polygon a 'while' loop was executed to apply a Rivara refinement algorithm to an initially coarse distribution based on the boundary set $\{p\}$. If an edge in the mesh is greater than the current maximum edge length, a new point is added to the mesh at the midpoint of that edge to split it in two.

In practice, the edge splitting is done using Rivara refinement [62, 63]. The resulting field is then smoothed using Laplacian smoothing in combination with Lawson flipping [39]. This process is repeated until all edges met the assigned target edge length, which could be a spatially variable field based on the distance to $\{\ell_{i,j}\}$, for example. While the resulting mesh quality is quite good, the process is inefficient and cumbersome.

The superior modeling qualities of variable resolutions can be made practical, if implementation complexities can be addressed. We do so here using a Poisson-disk sampling methodology where the final vertex distribution is directly created rather than iteratively derived. While the method was initially designed to specifically improve FRAM, we provide the details in a general format such that it can be implemented for any discrete fracture network methodology, including those that use both conforming and non-conforming flow and transport simulations. Details are given for Delaunay triangulations, which are of importance in many two-point flux finite volume solvers as they are used to generate the Voronoi control volumes on which these solvers compute. In the next section, we recount the properties of maximal Poisson-disk sampling that we used to design and implement this new method. Further we recount theoretical bounds on mesh gradation that ensure high-quality variable mesh resolutions.

2.4.2 Maximal Poisson-disk Sampling

Meshes from a sample that is dense yet cluster free have provable high quality bounds [7, 12, 16]. Similar quality bounds can be established for sphere-packings, whose radii are Lipschitz continuous with respect to their location [47, 48, 66]. Maximal, or almost maximal, Poisson-disk samplings fulfill all these conditions leading to high-quality meshes. Traditionally, Poisson-disk samplings are generated with an expensive dart-throwing algorithm [13]. These algorithms struggle to achieve maximality as the probability to select a free spot becomes decreasingly small. The algorithm in [49] based on these dart-throwing algorithm is the first to guarantee maximality and reaches run times of $O(n \log(n))$ (n : number of points sampled) by using a regular grid for acceleration and sampling from polygonal regions in its second phase to achieve maximality. They report close to $O(n)$ performance in practice [15, 17, 49]. Prior to that, an algorithm not based on dart-throwing was proposed in [9], which while not guaranteeing maximality, showed linear performance in the number of nodes sampled. Their algorithm was extended to variable radii in [14]. Other authors further provide algorithms that produce variable Poisson-disk samplings on 3D-surfaces[23],[24]. Their triangulation-based algorithm runs in linear time, and while theoretically not guaranteeing maximality, their experimental results suggest that maximality is achieved. A summary of relatively recent developments in this area along with comparison of different methods can be found in [69]. added references mentioned by reviewer 1, except for PushPull , the algorithm in there starts with non PDS and starts moving points around. Since this is specifically what we're trying to avoid it doesn't seem suitable imo. Maximal Poisson-disk samplings X on a domain $\Omega \subseteq \mathbb{R}^d$ are random selections of points $X = \{\mathbf{x}_i\}_{i=1}^n$, that fulfill the following properties:

1. *empty disk property*:

$$\forall i \neq j \in \{1, ..., n\} : |\mathbf{x}_i - \mathbf{x}_j| > r.$$

   We will call r the *inhibition radius*,

2. *maximality*:

$$\Omega = \bigcup_{i=1}^{n} B_R(\mathbf{x}_i),$$

where $B_\varepsilon(\mathbf{x}) = \{\mathbf{y} \in \Omega : |\mathbf{x} - \mathbf{y}| < \mathbf{y}\}$ is the open ball of radius ε around $\mathbf{x}$. R will be called the *coverage radius*. [49]

Intuitively, the *empty disk property* says that every sample point is at the center of d-dimensional ball or disk that does not contain any other points of the sampling. *Maximality* implies that these balls cover the whole domain, i.e., there is no point $y \in \Omega$, that is not already contained in one of the balls around a point in the sample.

It is useful to generalize these definitions, such that both the *inhibition* and the *coverage radius* depend on the sampling points, i.e. $r = r(\mathbf{x}_i, \mathbf{x}_j)$ and $R = R(\mathbf{x}_i, \mathbf{x}_j)$ for all $\mathbf{x}_i, \mathbf{x}_j \in X$. We hereon refer to this construct as a variable radii maximal Poisson-disk sampling, and we refer to a Poisson-disk sampling with constant radii as a fixed-radii maximal Poisson-disk sampling.[49]

A common approach is to assign each point $\mathbf{x} \in \Omega$ a positive radius $\rho(\mathbf{x})$ and have $r(\mathbf{x}_i, \mathbf{x}_j)$ be a function of $\rho(\mathbf{x}_i)$ and $\rho(\mathbf{x}_j)$. Natural choices for $r(\mathbf{x}_i, \mathbf{x}_j)$ are, for example, $\rho(\mathbf{x}_i)$ or $\rho(\mathbf{x}_j)$ for $i < j$, thereby determining the inhibition radius depending on the ordering on X. Order independent options include $\min(\rho(\mathbf{x}_i), \rho(\mathbf{x}_j)), \max(\rho(\mathbf{x}_i), \rho(\mathbf{x}_j))$ or $\rho(\mathbf{x}_i) + \rho(\mathbf{x}_j)$. The last of these options corresponds to a sphere packing [49]. The coverage radius can, but does not have to be different from ρ.

The Delaunay triangulation of a sampling maximizes the smallest angle of its triangles among all triangulations of this sampling [43]. Since numerical errors in many applications tend to increase if these angles become smaller [71], Delaunay triangulations often are a triangulation of choice. Moreover, the dual of the Delaunay triangulation is a Voronoi tessellation, which in a certain sense is optimal for two-point flux finite volume solvers [19], that are commonly used in subsurface flow and transport simulators such as FEHM [72], TOUGH2 [61], and PFLOTRAN [41]. In case of maximal Poisson-disk samplings we can go one step further and give a lower bound on these angles. In what follows we estimate the bounds that apply to the sampling we generate on DFN in later sections. We provide a brief summary of the proofs found in [49], while highlighting the most important results we use. We then proceed with the new bounds.

Lemma 2.1. *The smallest angle α in any triangle is grater than* $\arcsin\left(\frac{r}{2R}\right)$*, where r is the length of the shortest edge and R the radius of the circumcircle or*

$$\sin(\alpha) \geq \frac{r}{2R} \tag{2.1}$$

Proof. This is a direct corollary of the central angle theorem. □

This Lemma allows us to give explicit bounds for maximal Poisson-disk samplings. While we will focus entirely on inhibition radii given by $r(\mathbf{x}_i, \mathbf{x}_j) = \min(\rho(\mathbf{x}_i), \rho(\mathbf{x}_j))$, where $\rho(\mathbf{x})$ is some positive function, comparable results can be found for different $r(\mathbf{x}_i, \mathbf{x}_j)$ in a similar fashion.

Lemma 2.2. *Let $\varepsilon \geq 0$ and $\rho : \mathbb{R}^n \to \mathbb{R}$ $(n \geq 2)$ be a positive Lipschitz continuous function with Lipschitz constant L with $L\varepsilon < 1$. Let $X \subset$ be a variable maximal Poisson-disk sampling on the domain $\Omega \subset \mathbb{R}^n$ with inhibition radius $r(\mathbf{x}, \mathbf{y}) = \min(\rho(\mathbf{x}), \rho(\mathbf{y}))$ and coverage radius $R(\mathbf{x}, \mathbf{y}) \leq (1 + \varepsilon) r(\mathbf{x}, \mathbf{y})$.$(\varepsilon > 0)$*
Let the triangle Δ be an arbitrary element of the Delaunay triangulation of X $(n = 2)$ or an arbitrary 2-dimensional face of a cell of the Delaunay triangulation of X.
If the circumcenter of Δ is contained in Ω, each angle α of Δ is greater or equal to $\arcsin\left(\frac{1-L-\varepsilon L}{2+2}\varepsilon\right)$ *or*

$$\sin(\alpha) \geq \frac{1 - L - \varepsilon L}{2 + 2\varepsilon}.$$

Proof. Let α be the smallest angle of Δ and $\mathbf{x}, \mathbf{y} \in X$ be the vertices of the shortest edge of Δ, i.e. the vertices opposite to α. Without loss of generality assume $\rho(\mathbf{x}) \leq \rho(\mathbf{y})$. Since X is a Poisson-disk sampling $|\mathbf{x} - \mathbf{y}| \geq \min(\rho(\mathbf{x}), \rho(\mathbf{y})) = \rho(\mathbf{x})$.
Now let $\mathbf{z} \in \Omega$ be the circumcenter of Δ. Since X is maximal, there exists $\mathbf{v} \in X$ with $|\mathbf{z} - \mathbf{v}| \leq R(\mathbf{z}, \mathbf{v}) \leq (1 + \varepsilon)\rho(\mathbf{z})$. Next we notice that, because Δ was retrieved from a Delaunay triangulation $\mathbf{v}$ cannot be contained in the interior of Δ's circumcircle. Hence

$$|\mathbf{z} - \mathbf{x}| \leq |\mathbf{z} - \mathbf{v}| \leq (1 + \varepsilon)\rho(\mathbf{z}) \leq (1 + \varepsilon)\left(\rho(\mathbf{x}) + L|\mathbf{z} - \mathbf{x}|\right).$$

Rearranging this inequality yields

$$|\mathbf{z}-\mathbf{x}| \leq \rho(\mathbf{x})\frac{1+\varepsilon}{1-L-\varepsilon L}.$$

The result follows by applying Lemma 2.1 after noticing that $|\mathbf{x}-y|$ is the length of the shortest edge and that $|z-\mathbf{x}|$ is the radius of the circumcirle. □

Remark 2.4.1. *Note that for $n > 2$ the same result is true, if we assume the circumcenter of the n-simplex, of which Δ is a face, is contained in Ω instead of the circumcenter of Δ itself. The proof is identical.*

Remark 2.4.2. *While this result allows to control the quality of 2D-triangulations of maximal Poisson-disk samplings, it can also be used to gauge how close a given Poisson-disk sampling is to being maximal.*

The previous Lemma only gives us bounds on all triangles, if their circumcenters are contained in Ω. The next two Lemmas will give sufficient conditions to guarantee exactly this as long as Ω is a polytope.

Lemma 2.3. *Let $\Omega \subset \mathbb{R}^2$ be a polygonal region and X a maximal Poisson-disk sampling containing all vertices of Ω. Let the inhibition radius $r(\mathbf{x},\mathbf{y})$ be defined like in the previous lemma. Further let the coverage radius of $X \cap \delta\Omega$ fulfill $R^\delta(\mathbf{x},\mathbf{y}) < \frac{r(\mathbf{x},\mathbf{y})}{\sqrt{2}(1+L)}$, i.e. $|\mathbf{x}-\mathbf{y}| < \frac{\sqrt{2}}{1+L}r(\mathbf{x},\mathbf{y})$ for all $\mathbf{x},\mathbf{y} \in \delta\Omega \cap X$, the circumcenter of all triangles in the Delaunay triangulation of X are contained in $\bar{\Omega}$.*

Proof. Suppose this claim is wrong. Then let Δ be a triangle in the Delaunay triangulation with circumcenter $z \notin \Omega$. For this to be possible the circumcircle needs to be cut in (at least) two pieces by $\delta\Omega$, separating z and the vertices of Δ. Since Δ is part of a Delaunay triangulation and all vertices of Ω are part of the sampling, this is done by (at least) one segment of a straight line, i.e. $\delta\Omega$ contains a secant of the circumcircle.
Let $\mathbf{b}_1, \mathbf{b}_2 \in \delta\Omega$ be the two boundary points closest to the circumcircle on either side of that line segment and let B be the disk bounded by the circumcircle. Note that $\bar{B} \cap \Omega$ contains Δ and is itself entirely contained in the disk of radius $\frac{1}{2}|\mathbf{b}_1-\mathbf{b}_2| < \frac{1}{2}\frac{\sqrt{2}}{1+L}r(\mathbf{b}_1,\mathbf{b}_2)$ around

$\frac{1}{2}(\mathbf{b}_1 + \mathbf{b}_2)$.

Now let $\mathbf{x} \notin \{\mathbf{b}_1, \mathbf{b}_2\}$ be a vertex of Δ and let $\mathbf{b} \in \{\mathbf{b}_1, \mathbf{b}_2\}$ be the point of the two, that is closer to $\mathbf{x}$. We already established that $\mathbf{x}$ lies within the just mentioned ball around $\frac{1}{2}(\mathbf{b}_1 + \mathbf{b}_2)$. Let $\mathbf{x}_p$ be the projection of $\mathbf{x}$ onto the line segment connecting b_1 and b_2. Then $|\mathbf{x}-\mathbf{x}_p| < \frac{1}{2}\frac{\sqrt{2}}{1+L} r(b_1, b_2)$, because $\mathbf{x}$ lies within the circle of that radius, $|\mathbf{x}_p - b| < \frac{1}{2}\frac{\sqrt{2}}{1+L} r(b_1, b_2)$, because b is the closer of the two points b_1, b_2 and therefore

$$|\mathbf{x} - \mathbf{b}| = \sqrt{|\mathbf{x} - \mathbf{x}_p|^2 + |\mathbf{b} - \mathbf{x}_p|^2} < \frac{r(\mathbf{b}_1, \mathbf{b}_2)}{1+L} \leq \frac{\rho(\mathbf{b})}{1+L} \leq \rho(\mathbf{b}). \tag{2.2}$$

Since $|\mathbf{x} - \mathbf{b}| \geq \min(\rho(\mathbf{x}), \rho(\mathbf{b}))$ this implies $|\mathbf{x} - \mathbf{b}| \geq \rho(\mathbf{x})$. However assuming this and applying the Lipschitz condition on (2.2) gives us

$$|\mathbf{x} - \mathbf{b}| < \frac{\rho(\mathbf{b})}{1+L} \leq \frac{1}{1+L}\left(\rho(\mathbf{x}) + L|\mathbf{x} - \mathbf{b}|\right) \leq \frac{1+L}{1+L}|\mathbf{x} - \mathbf{b}|,$$

which is a contradiction. □

Remark 2.4.3. *Lemma 2.3 does generalize to higher dimensions. It is not very practical because it is difficult to guarantee the bounds on R^δ, if the boundary is more than 1-dimensional. However it is still possible to get some bounds on the radii of the circumcircles and then, using Lemma 2.1, on the angles, if the distance of non-boundary nodes is greater than some lower bound $d > 0$.*

In fact, using notation from the previous proof, let Δ again be an n-simplex with circumcenter outside of Ω and $\mathbf{x} \notin \delta\Omega$ on of its nodes. Since the circumsphere of any simplex in a Delaunay triangulation does not contain any other nodes the radius of the intersection with $\delta\Omega$ is bounded by R^δ. One can show using simple geometric arguments that this forces the radius of Δ's circumsphere R to fulfill the following inequality

$$R^2 \leq (R-d)^2 + \left(R^\delta\right)^2 \Rightarrow R \leq \frac{d^2 + \left(R^\delta\right)^2}{2d} \leq \frac{\left(R^\delta\right)^2}{d}. \tag{2.3}$$

If $R^\delta(\mathbf{x},\mathbf{y}) < r(\mathbf{x},\mathbf{y})$ there is a lower bound on d, continuously depending on R^δ, solely due to the fact, that we have a Poisson-disk sampling. If $R^\delta = R^\delta(\mathbf{x}_p, b)$ is any bigger, d needs to be bounded artificially. This implies that the angle bounds change continuously, if the conditions for Lemma 2.3 cannot be met they still can be relatively controlled by the choice of the artificial bound on d.

Under the conditions of the previous Lemmas the simplices of the Delaunay triangulation are guaranteed to only have well-behaved triangular faces. In three or more dimensions however this does not imply that the simplices themselves are well-behaved. It is still possible for a Delaunay triangulation to contain slivers for example, that is tetrahedra whose 4 nodes are all positioned approximately on the equator of their circumsphere. In [11] slivers are characterized as tetrahedra, whose nodes are all close to a plane and whose orthogonal projection onto that plane is a quadrilateral. In [3] slivers are equivalently classified as tetrahedra with a dihedral angle close to 180° containing their own circumcenter. Slivers can have all their faces be equilateral triangles, yet have dihedral angles that are arbitrarily small, causing numerical errors to blow up.

While slivers cannot be entirely avoided, one can show that if the nodes $\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w}$ of a maximal Poisson-disk sampling form a sliver, the distance between $\mathbf{w}$ and the plane spanned by $\mathbf{x},\mathbf{y},\mathbf{z}$ needs to be very small [11]. This allows us to avoid slivers within certain planes, by first generating a 2D sampling in these planes and then enforcing a minimal distance between the plane and further nodes in the 3D sampling. We use this to avoid slivers around the DFN and the faces of the surrounding matrix. This also causes slivers to be rather scarce in a 3D maximal Poisson-disk sampling as given any three nodes the vast majority of possible positions of a fourth node do not produce a sliver. This scarcity of slivers in a sampling makes it quite likely that if nodes of slivers are removed and resampled the resulting triangulation will have less slivers than the previous one. This opens the door for a rejection-style algorithm to be successful in improving the overall quality of a triangulation.

2.5 Methods

Our proposed method for mesh generation is broken into three primary steps. First, we generate a 2D point distribution on each fracture in the DFN. After merging these samples and removing conflicts with regards to the empty disk property, we generate a 3D-Poisson disk sampling on the surrounding matrix by adding points wherever maximality allows it. Finally, in an attempt to remove slivers, we remove their nodes and randomly replace them until no slivers remain.

2.5.1 2D Sampling Method

We generate 2D Poisson-disk samplings in a successive manner using a rejection method. This method can be performed on every fracture in the network independent of the other fractures. (Details can be found in [30].) In each step a new candidate is generated, and if it does not break the empty disk-property with any of the already accepted nodes, it is accepted. For the sampling in two dimensions, we use a variable inhibition radius that increases linearly based on the distance to the closest intersection of the DFN.

In particular, we reject a candidate node $\mathbf{y}$, if there is an already excepted node $\mathbf{x}$ such that the condition

$$|\mathbf{x}-\mathbf{y}| \geq r(\mathbf{x},\mathbf{y}) = \min(\rho(\mathbf{x}),\rho(\mathbf{y})) \tag{2.4}$$

is violated. In this equation $\rho(\mathbf{x})$ as a piecewise linear function given by

$$\rho(\mathbf{x}) = \rho(D(\mathbf{x})) = \begin{cases} \frac{H}{2} & \text{for} \quad D(\mathbf{x}) \leq FH \\ A(D(\mathbf{x}) - FH) + \frac{H}{2} & \text{for} \quad FH \leq D(\mathbf{x}) \leq (R+F)H \\ (AR + \frac{1}{2})H & \text{otherwise} \end{cases}. \tag{2.5}$$

Here $D(\mathbf{x})$ is the Euclidean distance between $\mathbf{x}$ and the closest intersection. H, A, R and F are parameters, that determine the global minimal distance between two nodes ($H/2$), the range around an intersection on which the local inhibition radius remains at its minimum (FH), the global maximal inhibition radius ($ARH + H/2$), and the slope at which the inhibition

radius grow with $D(\mathbf{x})$ (A). Since $\rho(D)$ is piecewise linear, it is a Lipschitz-function with Lipschitz-constant A.

If the sampling has a coverage radius $R(\mathbf{x}, \mathbf{y}) \leq (1+\varepsilon)r(\mathbf{x}, \mathbf{y})$ for some $\varepsilon > 0$ the conditions of (2.2) hold. To satisfy the conditions of (2.3) as well and thereby ensure angle bounds on all triangles in a Delaunay triangulation we first sample along the boundary, enforcing a maximal distance of $\frac{r(\mathbf{x},\mathbf{y})}{\sqrt{2}(1+L)}$ between boundary nodes. As shown in [9] and [14], we generate new candidates for our sampling randomly on an annulus around an already accepted node. This is illustrated in Figure 2.1. The inner radius of this annulus is determined by the minimal distance another node could have to the center node, while still preserving the empty disk property, whereas the outer radius is determined by the maximal distance a node could have to the center in a maximal sampling. For our choice of inhibition radius, assuming the same radius as coverage radius, these distances can be made out to be $\frac{\rho(\mathbf{x})}{1+A}$ for the inner radius and $\frac{2\rho(\mathbf{x})}{1-A}$ for the outer one.

We will now go over the individual steps of the 2D algorithm. These steps can also be found in the pseudocode Algorithm 1 in Section 2.5.3 and are illustrated in figure 2.3. The necessary notation to read the pseudocode is found in the table at the start of the same section. In line 3 of that code a 1D Poisson-disk sampling along the boundary of the polygon is generated as a seed to start the algorithm. We continue to sample k new candidate nodes at a time (line 13) around each already accepted node and determine whether they get accepted or not (lines 14 through 22). k is a positive integer and a user-defined parameter of the algorithm. If all k candidates around a node are rejected, we move on to the next already accepted node (line 30). The algorithm terminates for the first time as soon as every accepted node was the sampling center once(line 31). Following [9] and [14], we use cell-lists to find nodes around a candidate that could potentially cause this candidate to violate the empty-disk property, as depicted in Figure 2.2(a). The size of these cells is chosen to contain at most one node. This allows us to disregard distance calculation with nodes beyond a certain cutoff and therefore allows us to achieve linear run times in the number of generated nodes (line 17). However, unlike the previously mentioned algorithms we do not only label cells containing particles as occupied, but also cells that are too close to an accepted node to contain a particle. In particular, if a candidate $\mathbf{x}$ lies in a cell C and any

other cell D with $diam(C \cup D) \leq r_{in}(\mathbf{x})$ is occupied, $\mathbf{x}$ can be rejected right away as it conflicts with the node in D (line 14). On the other hand, if $dist(C, D) > \rho(\mathbf{x})$, there is no need to calculate the distance between $\mathbf{x}$ and any potential element of D, as they can never violate the empty disk-property. An example of that is shown in Figure 2.2(b). We use this to our advantage in two ways: First, it allows us to reject many candidates without calculating any distances to nearby nodes, which particularly for large values of k gives a respectable speedup compared to the original algorithm; second, unmarked cells are easy to find and contain at least some space for another node, allowing us to find undersampled regions after the algorithm terminated(line 2). We fill these holes in the sample by generating random candidates within these unmarked cells(line 3). The main algorithm is then restarted from these newly added nodes until it terminates again (line 16). While this process can be repeated several times, just a single resampling already increases the quality of the sampling tremendously.

Once the point distribution is created, the conforming Delaunay triangulation method of [51] is used to create the final mesh on the fracture. In order for a conforming Delaunay triangulation which preserves the lines of fracture intersections as a set of triangle edges to be created, it is sufficient that the circumscribed circle of each segment of the discretized line of intersection be empty of any other node in the point distribution prior to connecting the mesh. To achieve this condition, any node within the circumscribed circle of each segment of the discretized lines of intersection is removed from the point distribution. Next, a 2-dimensional unconstrained Delaunay triangulation algorithm is used to connect this node set. Because of the construction method, i.e., empty regions around the lines of intersection, the line segments that represent lines of fracture intersection must emerge in the triangulation and the Delaunay triangulation will conform to all of the fracture intersection line segments. Once every fracture polygon is triangulated, they are all joined together into a unified triangulated fracture network.

2.5.2 3D sampling method

The sampling in 3D works very similar to its 2D counterpart. However, new candidates are generated on a spherical shell around accepted nodes instead of on an annulus. The 3D

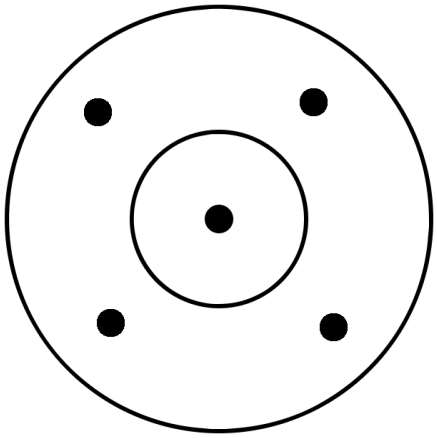


Figure 2.1: Visualisation of a single sampling step. Current node at center, new candidates in annulus (k=4). Inner circle is bounded by the inhibition radius of the current node. Outer circle is bounded by maximal distance a node could be away from the current if the Poisson-disk sampling was maximal. (In text mentions: p.20)

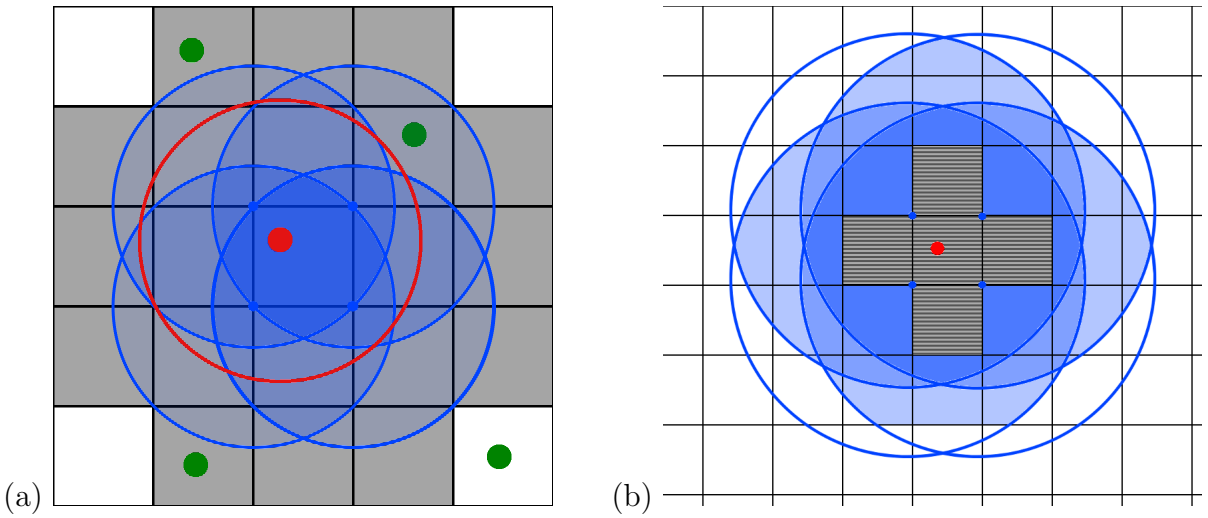


Figure 2.2: Visualisation of how the grid is used to find possibly conflicting nodes. New candidate in red, already accepted nodes in green, cells that can contain conflicting nodes in grey. Red circle shows the inhibition radius of the candidate, blue circles show furthest cells a node in the center cell could conflict with. (In text mentions: pp.20,21)

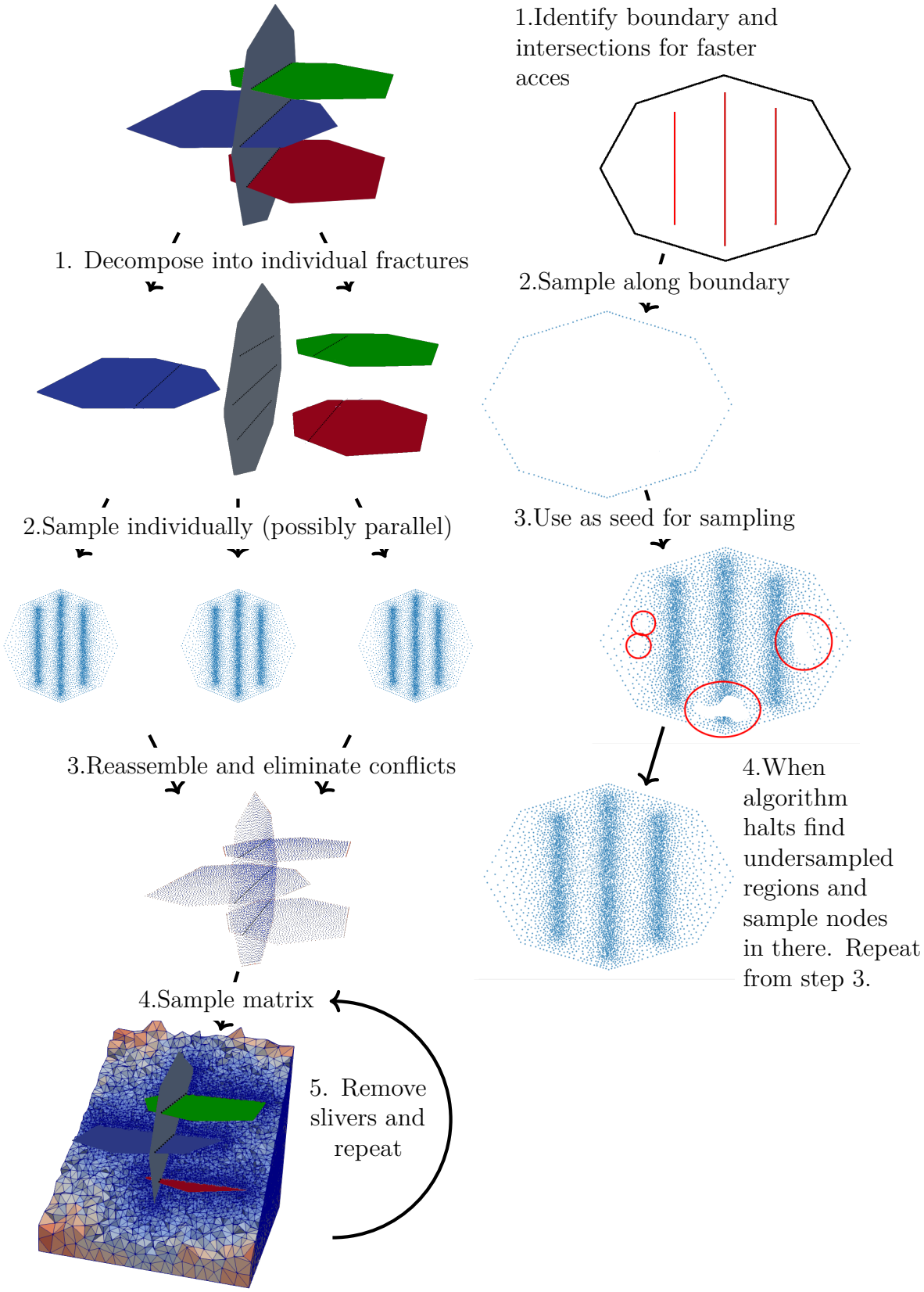


Figure 2.3: Overview of workflow between creation of DFN and final mesh (left) and overview of workflow during 2D-sampling (right). (In text mentions: pp.20,25)

variant of $\rho(\mathbf{x})$ given by

$$\rho(\mathbf{x}) = \rho(D(\mathbf{x})) = \begin{cases} \rho_2(\mathbf{x}_p) & \text{for} \quad D(\mathbf{x}) \leq F\rho_2(\mathbf{x}_p) \\ A(D(\mathbf{x}) - Fr_{2D}(\mathbf{x}_p)) + \frac{H}{2} & \text{for} \quad F\rho_{2D}(\mathbf{x}_p) \leq D(\mathbf{x}) \leq \frac{\rho_{max} - r_2(\mathbf{x}_p)}{A} \\ \rho_{max} & \text{otherwise} \end{cases}. \tag{2.6}$$

$\mathbf{x}_p$ the fracture point closest to $\mathbf{x}$ and $\rho_2(\mathbf{x}_p)$ is its 2D inhibition radius on the fracture. $D(\mathbf{x})$ is the distance between $\mathbf{x}$ and $\mathbf{x}_p$. Like in 2D, this is a piecewise linear function in $D(\mathbf{x})$, which is constant, if within a distance of $\rho_2(\mathbf{x}_p)F$ (F a parameter) and then increases linearly with a slope of A until the maximal inhibition radius of ρ_{max} is reached. In addition to rejecting all candidates y for which (2.4) is violated, we also reject a candidate $\mathbf{x}$, if it is within a distance of $\rho(\mathbf{x})/2$ to a boundary or fracture. This both prevents slivers from having three nodes located on a single fracture or the boundary of the matrix and limits the circumradius of tetrahedra with circumcenter outside of the matrix (lemma 2.3 and subsequent remark).

A pseudocode of how the 3D-sampling is run from here can be found in Section 2.5.3 in Algorithm 2 . The necessary notation is listed in the table at the start of that section. As the first sampling process is essentially identical to the 2D version, we will explain the differences in the initialization and the resampling. At the start, the nodes are initialized through a Poisson-disk sampling on the boundary of the 3D matrix and the sampling on the DFN generated by the 2D algorithm (line 2). Neighbor cells can still be used in the same way as in 2D to speedup the rejection of candidates. Unlike in 2D, a maximal Poisson-disk sampling does not guarantee sliver-free triangulation, which is why we do not use the cell lists to find undersampled cells in 3D. Instead, once the algorithm terminates, the resulting sampling is triangulated (line:10), slivers identified (line:11), and 2 nodes of every sliver (with a preference for nodes, that are neither on a boundary or a fracture) removed (line 12). While the definition of a sliver given earlier in section 2.4.2 allows for a bit of leeway in what is considered a small or large dihedral angle, in practice we successfully replaced tetrahedra with dihedral angles outside of $[8°, 170°]$ and aspect ratios bigger than 0.2. Then the algorithm is

restarted with the remaining nodes as seed (line 15). This process is repeated til a sliver-free sampling is obtained(line 16). With this approach we have been able to obtain triangulations with no elements of dihedral angles of less than 8° (presented in next sections). The method for generating the conforming mesh is similar to that for the 2-dimensional case, but spheres around triangle cells of the fracture planes are excavated. Additional details are found in [36].

2.5.3 Workflow Overview & Pseudocode for the 2D and 3D sampling algorithms

The workflow is depicted in Figure 2.3 and contains the following high-level steps: (1) generation of a DFN using dfnWorks [34], (2) decomposition of DFN into individual polygons, (3) generation of 2D-variable-radii Poisson-disk samplings on each individual polygon using algorithm 1 below, (4) construct a conforming Deluanay triangulation as previously described, (5) merge individual fracture meshes into a sampling on the original DFN, removing conflicting nodes along intersections, (6) generating a conforming 3D variable radii Poisson-disk sampling of the surrounding matrix of the DFN using the 2D samplings as seed according to Algorithm 3, (7) triangulate sampling, identify low-quality tetrahedra and remove 2 of their nodes that are not located on the original DFN, (8) repeat steps 5 and 6 with the remaining nodes as seed until no more low-quality tetrahedra remain [22]. Replacing step (8) with more traditional ways of sliver-removal like perturbation [67] or exudation [11] can break the empty disk property of the sampling.

Notation for Pseudocodes:

Input:

- $D^3 \subset \mathbb{R}^3$: cubical domain $(*)$
- $DFN \subset D^3$: generated by DFNWorks $(*)$
- $F_l \subset \mathbb{R}^3$: l-th fracture of the DFN
- $q_{l,m}^{(1)}$ and $q_{l,m}^{(1)}$: endpoints of intersection between fractures F_l and F_m

User defined parameters:

- $H/2$: minimal distance between nodes
- F : HF is range of constant density around intersections
- R: $ARH + H/2$ is maximal distance between nodes
- A: max. slope of inhibition radius
- k: number of concurrently sampled candidates

Additional notation:

- G: square cells covering F_l with $diam(g) \leq H/2$ for all $g \in G$.
- $\rho(\mathbf{x})$: as defined in equation (2.5)(2D) or (2.6) (3D)
- $r(\mathbf{x}, \mathbf{y})$: inhibition radius $\min(\rho(\mathbf{x}), \rho(\mathbf{y}))$
- $R(\mathbf{x}, \mathbf{y})$: coverage radius
- $C(\mathbf{x}) \in G$: grid cell containing the point x.
- $N^+(\mathbf{x})$: $\{g \in G : dist(C(\mathbf{x}), g) \leq \rho(\mathbf{x})\}$: cells that can contain points y with $|\mathbf{x} - y| \leq r(\mathbf{x}, y)$
- $N^-(\mathbf{x})$: $\{g \in G : diam(g \cup C(\mathbf{x})) \leq \frac{\rho(\mathbf{x})}{1+A}\}$: cells, where for all their points y $|\mathbf{x} - y| \leq r(\mathbf{x}, y)$
- G_{occ} $\bigcup_{\mathbf{x} \in X} N^-(\mathbf{x})$: cells on which X is already maximal.
- $\mathcal{T}(X)$: Delaunay triangulation of X $(*)$

Output:

- X : Poisson-disk sampling on the l-th fracture

$(*)$: 3D only

Algorithm 1 2D Poisson-disk sampling

Initializing:
$X \subset F_l$ ▷ Generate a 1D Poisson-disk sampling with $R(\mathbf{x}, y) \leq \frac{r(\mathbf{x},y)}{\sqrt{2}(1+L)}$ along boundary δF_l as seed.
for $\mathbf{x} \in X$ **do**
    $G_{occ} \leftarrow G_{occ} \cup N^-(\mathbf{x})$ ▷ Initialize occupied cells
end for

Sampling:
$i \leftarrow 1$ ▷ Start sampling at first accepted node.
$N \leftarrow |X|$ ▷ Will increase as more nodes are accepted
while $i \leq N$ **do**
    repeat
        for $j \in \{1, ..., k\}$ **do**
            $\mathbf{p}_j \in F_l$ ▷ Generate k new candidate nodes on the annulus around $\mathbf{x}_i$
            if $C(\mathbf{p}_j) \in G_{occ}$ **then**
                reject $\mathbf{p}_j$ ▷ Cell already blocked by existing node's inhibtion radius
            else
                for $\mathbf{y} \in N^+(\mathbf{p}_j)$ **do**
                    if $|\mathbf{p}_j - \mathbf{y}| < r(\mathbf{p}_j, \mathbf{y})$ **then**
                        reject $\mathbf{p}_j$ ▷ Empty disk property violated
                    end if
                end for
            end if
            if p_j was not rejected **then**
                $X \leftarrow X \cup \{p_j\}$ ▷ Accept $\mathbf{p}_j$ and add it to the sampling
                $G_{occ} \leftarrow G_{occ} \cup N^-(\mathbf{p}_j)$ ▷ Update occupied cells
                $N \leftarrow N + 1$ ▷ Ensures sampling around newly accepted nodes
            end if
        end for
    until All k of the $\mathbf{p}_j$ are rejected
    $i \leftarrow i + 1$ ▷ Start sampling around next accepted node
end while ▷ Terminate here or start resampling

Continuation of Algorithm 1 (2D Resampling)

Resampling:(optional: algorithm terminates, if no resampling is required.)
for $C \in G \setminus G_{occ}$ **do**
  $\mathbf{p} \in C$ ▷ Generate a random candidate on each cell
  for $\mathbf{y} \in N^+(\mathbf{p})$ **do**
    if $|\mathbf{p} - \mathbf{y}| < r(\mathbf{p}, \mathbf{y})$ **then**
      reject $\mathbf{p}$ ▷ Empty disk property violated
    end if
  end for
  if $\mathbf{p}$ was not rejected **then**
    $X \leftarrow X \cup \{\mathbf{p}\}$ ▷ Accept $\mathbf{p}$ and add it to the sampling
    $G_{occ} \leftarrow G_{occ} \cup N^-(\mathbf{p})$ ▷ Update occupied cells
    $N \leftarrow N + 1$
  end if
end for

Rerun algorithm again from line 10 (i is not reset.)

Algorithm 3 3D Poisson-disk sampling + Resampling

Initializing:
$X \subset D^3$ ▷ Use Algorithm 1 to generate a Poisson-disk sampling on δD^3 and the DFN by using Algorithm 1 (remove conflicting node, when merging samplings.)
for $\mathbf{x} \in X$ **do**
  $G_{occ} \leftarrow G_{occ} \cup N^-(\mathbf{x})$ ▷ Initialize occupied cells
end for
Sampling:
The sampling process in 3D works exactly like in 2D with the two only difference being the following:

- New candidates are generated on a spherical shell instead on an annulus
- A candidate $p \notin \delta D^3$ is rejected if $dist(p, \delta D^3) < \rho(p)/2$

Resampling: (optional: algorithm terminates, if no resampling is required.)
repeat
  for $T \in \mathcal{T}(X)$ **do**
    if T is a sliver **then**
      $X \leftarrow X \setminus \{\mathbf{x}, \mathbf{y}\}$, where $\mathbf{x}, \mathbf{y} \in T$ are 2 random nodes not contained in the boundary or the DFN
      ▷ Minimal distance of nodes to DFN and boundary assures, that this is possible.
    end if
  end for
  Rerun algorithm again from line 6
until $\mathcal{T}(X)$ contains no more slivers.

2.6 Results

The following sections will present and discuss numerical results in 2D and 3D.

2.6.1 Two-dimensional Examples

Figure 2.4 shows the triangulation of a variable radius sampling on a simple fracture with 3 intersections. Triangles are colored by their maximal edge length showing how the triangle size increases as we move further away from the intersections.

In Figure 2.5, we depict the triangulation of a constant-radius sampling on that same fracture, put back together into the original DFN it originated from. This process does not influence the overall triangulation quality unless the fractures themselves intersect in an angle smaller than the angles of triangles in the triangulation.

We show an example from a slightly bigger DFN combining both variable radii Poisson-disk sampling and the reassembly into its original form in Figure . The network contains 25 fractures whose radii are generated from an exponential distribution with decay exponent of 0.3. There are up to eight intersections on each fracture, but note this is not a constraint of generation or the sampling technique. The parameters of the inhibition radius are set to $H = 0.1$,$A = 0.1$,$F = 1$ and $R = 40$.

The high quality of this particular triangulation is showcased in the histograms in Figure 2.7. Depicted are the distribution of minimal angles (a), maximal angles and the aspect ratios of the triangulation. We see one triangle each with 25° and 26° respectively as minimal angles with all other minimal angles being greater than 27°. The theoretical minimum angle in a maximal Poisson-disk sampling with Lipschitz constant $A = 0.1$ is 27.04°. The majority of minimal angles is significantly better still. In terms of the maximal angle, we can observe very few triangles with angles worse than 110° and none worse than 120°. The greatest maximal angle theoretically possible in a maximal Poisson-disk sampling with this Lipschitz-constant would be 125.92°. The vast majority of aspect ratios is greater than 0.8 with only a marginal number of triangles having an aspect ratio of less than 0.6 and none below 0.47.

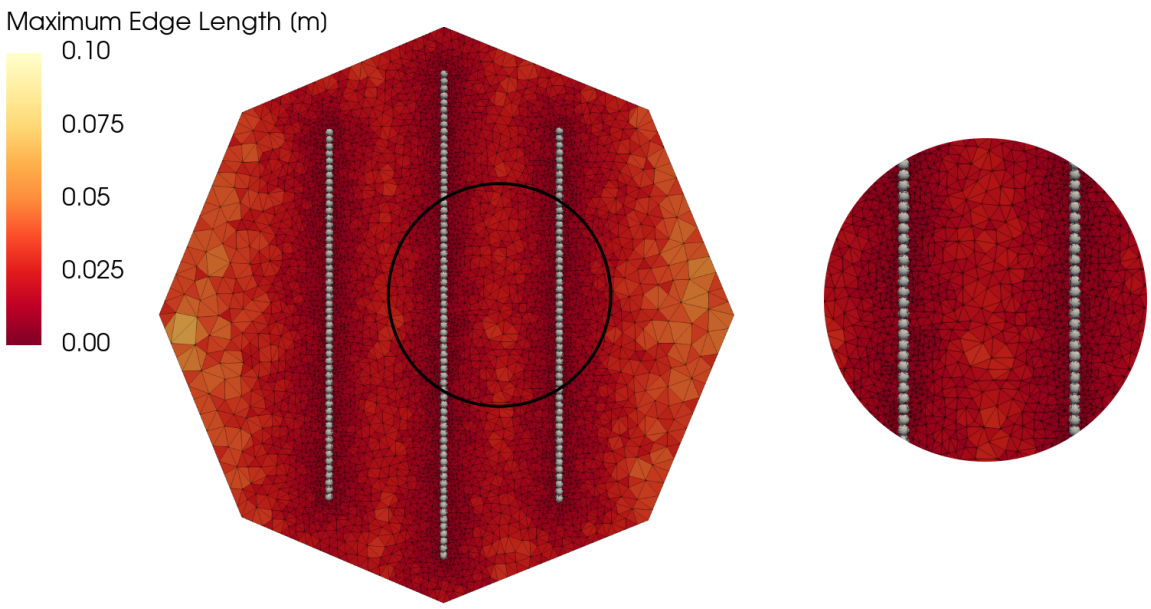


Figure 2.4: Triangulation of variable radii Poisson-disk sampling on fracture with three intersections. (H=0.01, R=40, A=0.1, F=1) Triangles colored according to their maximal edge length. The lines of intersection are shown as spheres. (In text mentions: p.29)

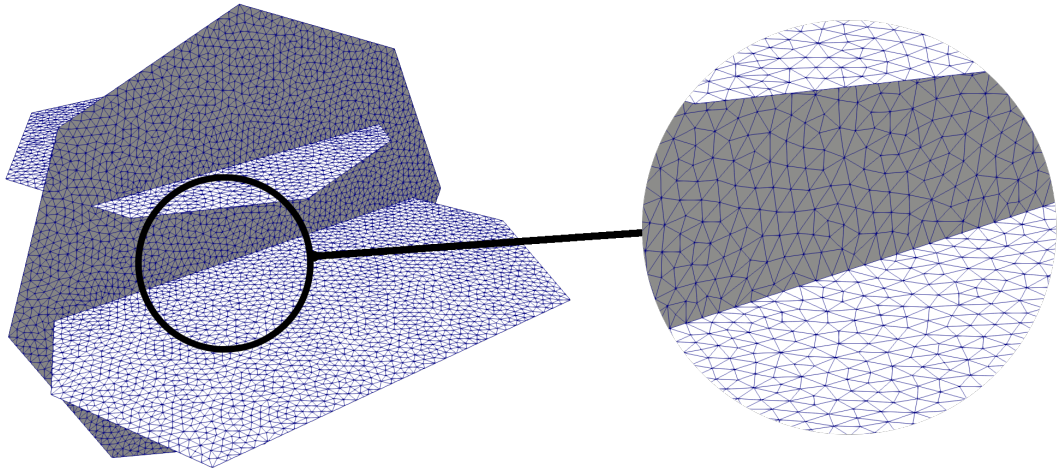


Figure 2.5: Triangulation of a regular Poisson-disk sampling reassembled into the original DFN. (In text mentions: p.29), (In text mentions: p.32)

2.6.2 Run Time Analysis

We show an analysis of the run time and quality of the sampling on a DFN for varying sample sizes, variations of the parameter k, and different numbers of resampling attempts. All these data points were generated on the same DFN. Different node numbers were achieved by continuously changing the parameter H, the minimal allowed distance between nodes. All data points are from independent samplings. The plot in Figure 2.8 shows the run time prior to resampling process against the number of nodes sampled up to that point. The color corresponds to the value of the parameter k, which controls the number of concurrent samples. We see an increase in run time with increasing k, as expected. The run times for samples with the same k are positioned along straight lines of slope one, indicating a linear dependence of the total run time and the number of nodes sampled. The red lines in the plot have a slope of 1 to help visualize this. Figure 2.9 shows the relation between the parameter k and the run time. Colors correspond to different numbers of nodes. As already established, the run time increases linearly with the number of nodes sampled. The run time in terms of k even exhibits a slightly sublinear behavior. The linear fit (black) of the data in this log-log-plot has a slope of $0.7(9) \pm 0.00(7)$. While this fitting error of $\approx 9\%$ is not insignificant it can also clearly be seen by comparing the data to the two lines of slope 1 (red) in the plot, that the run time does not increase more than linearly with k.

Figure 2.10 depicts a comparison of runtime between our implementation of [14] or [9] for variable-radii sampling and the same implementation with our adaptation to use the grid not only to find closeby nodes, but also directly reject candidates. Data points generated by our adapted algorithm are represented by a filled circle, whereas data points generated by the original algorithm are shown by empty squares. All data points are colored depending on k. We can see our algorithm out performs the original for every pair of data points. This advantage increases with growing k, which makes sense as there are more rejected candidates the greater k is and our adapted version can handle rejection faster since it does not have to calculate the distance. For $k = 5$ the speed difference between the algorithms is slightly less than a factor of 2, whereas for $k = 160$ the advantage grows to about an order

of magnitude. Some comparisons to the sampling algorithm used in DFNWORKS [33] prior to this implementation are shown in table 2.1. This original implementation uses a Rivara refinement algorithm to generate the nodes of the mesh.

The speedup in run time is presented in Table 2.1. We consider three different DFN to characterize the difference between the methods. The first is the deterministic network of four ellipses shown in Fig. 2.5. The second is the network with fractures sampled from an exponential distribution containing 25 fractures shown in Fig. 2.6. The final network is composed of a single family of disc shaped fractures whose fracture lengths are sampled from a truncated powerlaw with exponent 1.8, minimum length 1 m, maximum length 25 m within a cubic domain wide sides of length 100 m. There are 8417 fractures in this network. In the first two examples, the mesh was run on a MacBook Pro laptop with 8 2.9 GHz Intel Core i9 processors and 32GB of RAM. The third example was run on a linux server with 64 AMD Opteron(TM) Processor 6272 (1469.697 MHz) and 252GB of RAM. The mesh resolution and setup were consistent between the methods. In all cases, the Poisson-Disk method was faster than the iterative method, and appears to improve in speed-up with number of fractures. However, the difference in network properties also plays a role in the speedup, a feature that we do not explore in this study.

2.6.3 Quality and resampling

The maximality of our samples correlates to a high degree to the choice of k, but also to the number of times the resampling algorithm is run. Depicted in Figure 2.11 are the total number of nodes sampled after a different number of resamplings. First we can see that the density of nodes grows with the parameter k. This growth starts out fast for small k and while not entirely ceasing to increase, slows down notably for higher k. (Note log-scale on **x**-axis.) On the lower end of the k scale, resampling increases the node density significantly, whereas there is barely any difference for higher $k > 100$. The first resampling is particularly effective, whereas the difference between each resampling decreases afterwards. Given that resampling does not take more time then the original sampling process, this turns into an interesting trade-off between higher k and more repetitions of the resampling that overall can yield higher performance. A run at $k = 5$ with few repetition for example, results in a

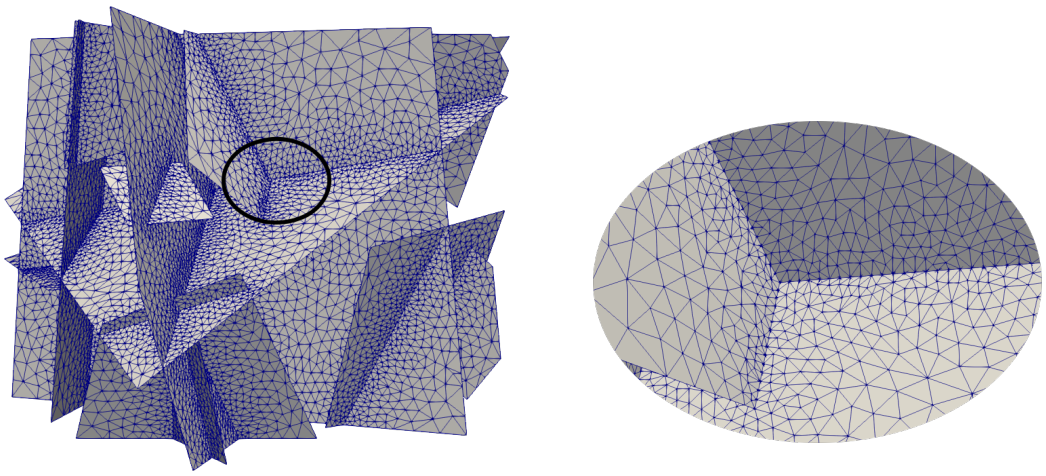


Figure 2.6: Triangulation of a variable radii Poisson-disk sampling reassembled into original DFN. The network contains 25 fractures whose radii are generated from an exponential distribution with decay exponent of 0.3. Parameters used in the sampled: $H = 0.1$,$R = 40$,$A = 0.1$,$F = 1$. The mesh contains 23195 nodes and 47367 triangles. The minimal angle is $\geq 25°$, maximal angle $\leq 120°$, and all aspect ratios are ≥ 0.47. (In text mentions: pp.29,32)

Table 2.1: Comparison of run time between previous iterative method with presented Poisson Disk method for mesh generation. In all cases the new method outperforms the iterative method. (In text mentions: pp.32,32)

DFN Description	Deterministic Ellipses	Exponential Distribution	Truncated Power-law
Number of Fractures	4	25	8417
Mesh Resolution	Uniform	Variable	Variable
Number of Processors	4	8	32
Iterative - Run Time	10.64 s	57.23 s	47.47 m
Poisson-Disk - Run Time	4.46 s	9.91 s	6.47 m
Speed-up	2.4	5.8	7.8

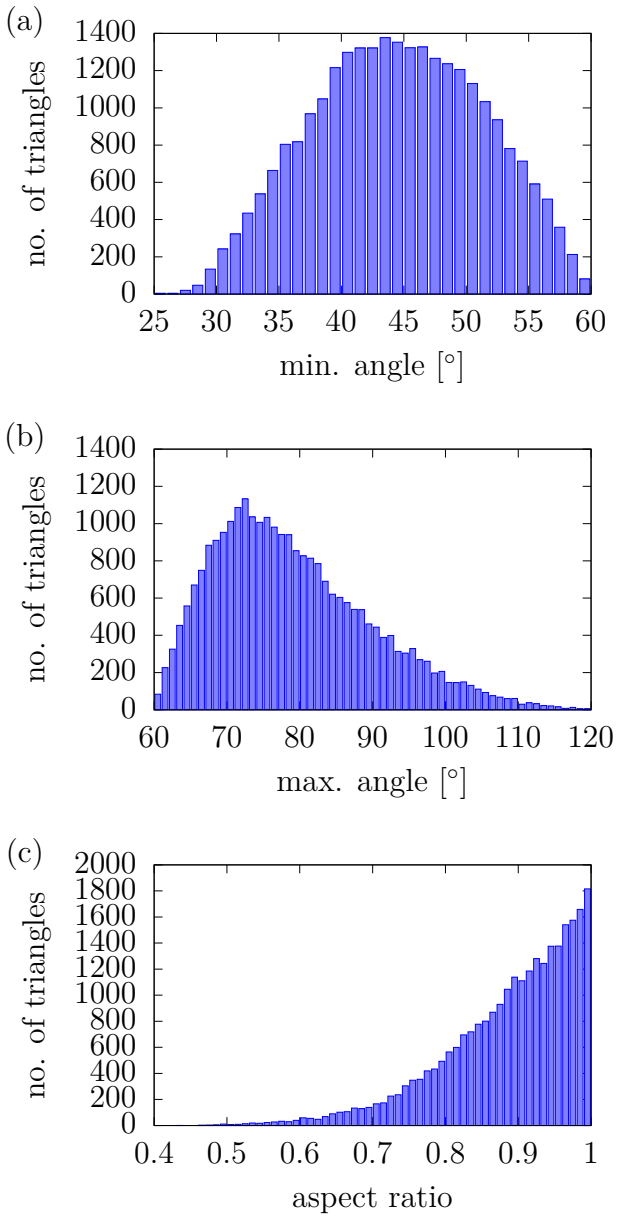


Figure 2.7: Histograms of selected quality measures of the triangulation of variable radii Poisson-disk sampling on a fracture with three intersections. (H=0.01, R=40, A=0.1, F=1). (a): minimal angle ($\geq 25°$), (b): max angle ($\leq 120°$), (c): aspect ratio (≥ 0.47) (In text mentions: p.29)

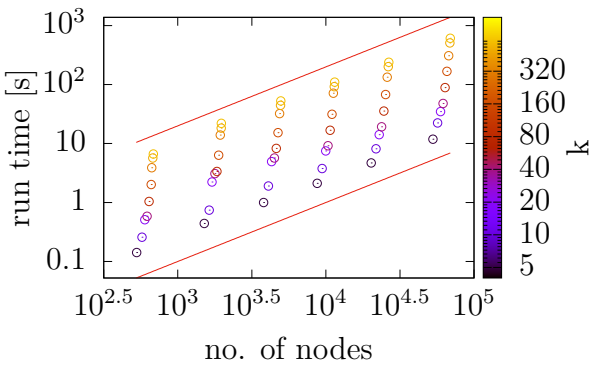


Figure 2.8: Log-log-plot of run time of Poisson-disk sampling algorithm in dependency of the total number of nodes sampled prior to the resampling process. Data points generated over the same DFN, different point densities generated by changing the minimal inhibition radius $\frac{H}{2}$ between every pair of nodes. Data points are colored depending on the value of k. Other parameters are set to $A = 0.1, R = 40, F = 1$. Comparison to lines of slope 1 (red) indicates the run time increases approximately at a linear rate. (In text mentions: p.31)

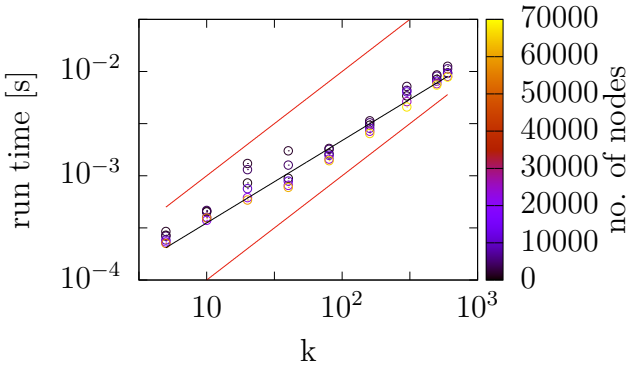


Figure 2.9: Log-log-plot of run time of Poisson-disk sampling algorithm in dependency of the number of concurrently sampled nodes k prior to the resampling process. Data points generated over the same DFN, different point densities generated by changing the minimal inhibition radius $\frac{H}{2}$ between every pair of nodes. Data points are colored depending on the total number of nodes sampled. Other parameters are set to $A = 0.1, R = 40, F = 1$. Linear fit (black) with slope $0.7(9) \pm 0.00(7)$. Comparison to lines of slope 1(red) indicate sublinear behavior. (In text mentions: p.31)

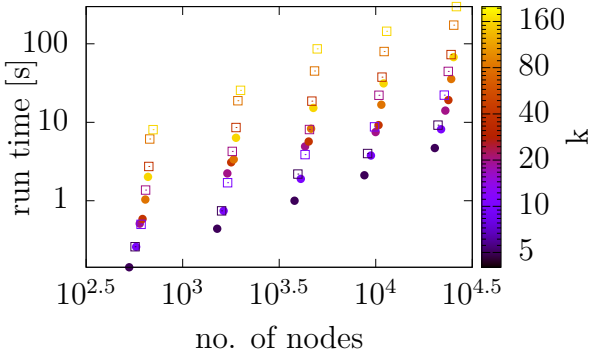


Figure 2.10: Comparison of run time for an implementation of [9, 14] (squares) and our variation ot the algorithm(circles). Depicted in a Log-log-plot are run time of Poisson-disk sampling algorithm in dependency of the total number of nodes sampled prior to the resampling process. Data points generated over the same DFN, different point densities generated by changing the minimal inhibition radius $\frac{H}{2}$ between every pair of nodes. Data points are colored depending on the value of k. Other parameters are set to $A = 0.1, R = 40, F = 1$. (In text mentions: p.31)

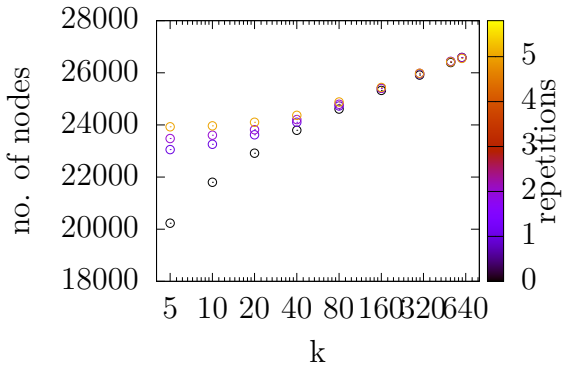


Figure 2.11: Total number of nodes sampled after resampling plotted in dependence of k colored by number of resamplings. Data points generated over the same DFN with fixed minimal inhibition radius. Other parameters are set to $A = 0.1, R = 40, F = 1$. (In text mentions: p.32)

density comparable to a run with more than 10 times higher k without resampling, while being significantly faster overall. Similar conclusions can be reached when looking at the quality of resulting triangulations rather than just the density of the Poisson-disk sampling.

Figure 2.12 shows the smallest minimal angle in a triangulation of our sampling for variable k and different numbers of resampling attempts. We can see for $k \gtrapprox 80$ this angle appears to be at around 25° independently of the number of repetitions. The theoretical bound for a maximal Poisson-disk sampling (with $r(\mathbf{x}, \mathbf{y}) = R(\mathbf{x}, \mathbf{y})$) for the settings used to generate these data points would be 27.04°. Solving the the angle bounds from Lemma 2.2 for ε shows us that in this sampling $R(\mathbf{x}, \mathbf{y}) \lesssim (1 + 0.1)r(\mathbf{x}, \mathbf{y})$. Given the statistical nature of the algorithm and the fact that identical inhibition and coverage radii are not quite guaranteed these results can be considered very good. While the quality of triangulations for smaller k without resampling is significantly lower, it is noteworthy that just a single repetition fixes this issue and yields triangulations with qualities on par with those for even significantly higher k. This allows the algorithm to run at single or low double digit k, perform a single resampling and generate a triangulation just as good as higher k would have produced in multitudes of the time.

2.6.4 Three-Dimensional Example

While the majority of our work was aimed at optimizing the 2D sampling on a DFN, we will conclude with an example where these 2D samplings are combined with a 3D sampling of the surrounding matrix to showcase that it can be used to produce high quality triangulations in this case as well. Triangulated output of the 3D algorithm can be seen in Figure 2.14. The tetradedra are colored according to their maximal edge length to show how the point density is adapted with the distance to the closest fracture.

Finally, the histograms in Figure 2.13 show the distribution of quality measures of the tetrahedra in the triangulation depicted in Figure 2.14. For this run, tetrahedra with either a dihedral angle of less than 8° or an aspect ratio of less than 0.2 were discarded before the sampling algorithm was restarted. The first histogram depicts the distribution of the minimal dihedral angle of each tetrahedron. As expected no dihedral angle below 8° remains, while the vast majority exceeds values of 30°. Histogram (b) shows that despite not optimizing with

respect to the maximal dihedral angle none of these angles exceed 165°. Histogram (c) shows a sharp cut-off at 0.2 in the distribution of aspect ratios indicating that the aspect ratio is likely to have been the driving factor for a majority of the resamplings. The example shown ran through a sliver-removal and resampling process 17 times to obtain its triangulation quality. In each of these steps a total of 200 or less out of approximately 50000 nodes were removed before the resampling. This showcases both the scarcity of slivers in samples generated through Algorithm 3, as well as that the vertices of these slivers can successfully be removed and replaced in a way that does not give rise to new slivers.

2.7 Conclusions

We considered 2 algorithms that successfully generate variable-radii Poisson-disk samples on polygonal regions or networks of polygons and the surrounding space they are embedded in.

Our experiments suggest that mesh quality is comparable in the method previously used in dfnWorks and the Poisson disk method we feature in this study, however, our algorithm produces comparable quality meshes in a significantly shorter time.

In our method high quality meshing results through the additional measures introduced to guarantee a significant degree of maximality. It is worth noting that maximality is reached for a coverage radius just slightly larger than the inhibition radius. Triangulations of these samplings show a quality almost matching theoretical quality bounds for maximal Poisson-disk samplings, in which coverage and inhibition radii coincide. Our key contributions are summarized as:

1. our algorithm is significantly faster than the previous conforming variable mesh strategies
2. for the fracture networks we achieved mesh quality only marginally worse than what is theoretically possible,
3. for the volume meshing, slivers can be removed entirely from the domain within certain bounds

It is worthwhile mentioning that the described algorithms are not only fast, but also simple to run in a parallel fashion, further improving the overall runtime. Given a DFN, the

2D-sampling can parallelized by working on each fracture on a different processor. Based on the grid structure used to accept and reject candidates, both 2D and 3D can also be further parallelized by dividing their domain into several pieces, that can be sampled individually on different processors, while needing to communicate only cell information on the boundaries of the split domains. Once these point distributions are produced, however, the all must reside on a single processor to connect them into a Delaunay mesh.

2.8 Acknowledgments

J.K. gratefully acknowledges support from the 2020 National Science Foundation Mathematical Sciences Graduate Internship to conduct this research at Los Alamos National Laboratory. J.D.H. and M.R.S. gratefully acknowledges support from the LANL LDRD program office Grant Number #20180621ECR, the Department of Energy Basic Energy Sciences program (LANLE3W1), and the Spent Fuel and Waste Science and Technology Campaign, Office of Nuclear Energy, of the U.S. Department of Energy. M.R.S. would also like to thank support from the Center for Nonlinear Studies. J.M.R. received support from DOE, Contract No. DE-AC05-00OR22725. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (Contract No. 89233218CNA000001). This work was prepared as an account of work sponsored by an agency of the United States Government. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, its contractors or subcontractors. LAUR # LA-UR-21-24804.

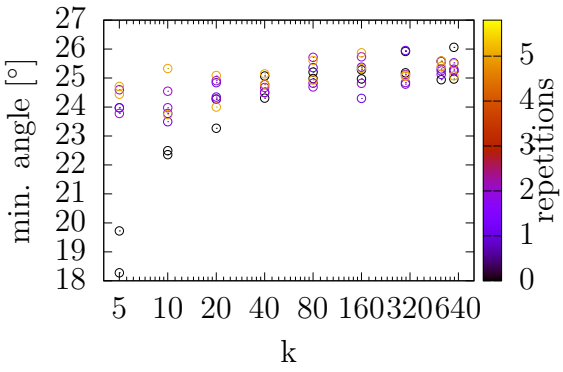


Figure 2.12: Smallest minimal angle of the triangulation of Poisson-disk samplings in dependence of k colored by the number of resamplings. Data points generated over the same DFN with fixed minimal inhibition radius. Other parameters are set to $A = 0.1, R = 40, F = 1$. (In text mentions: p.37)

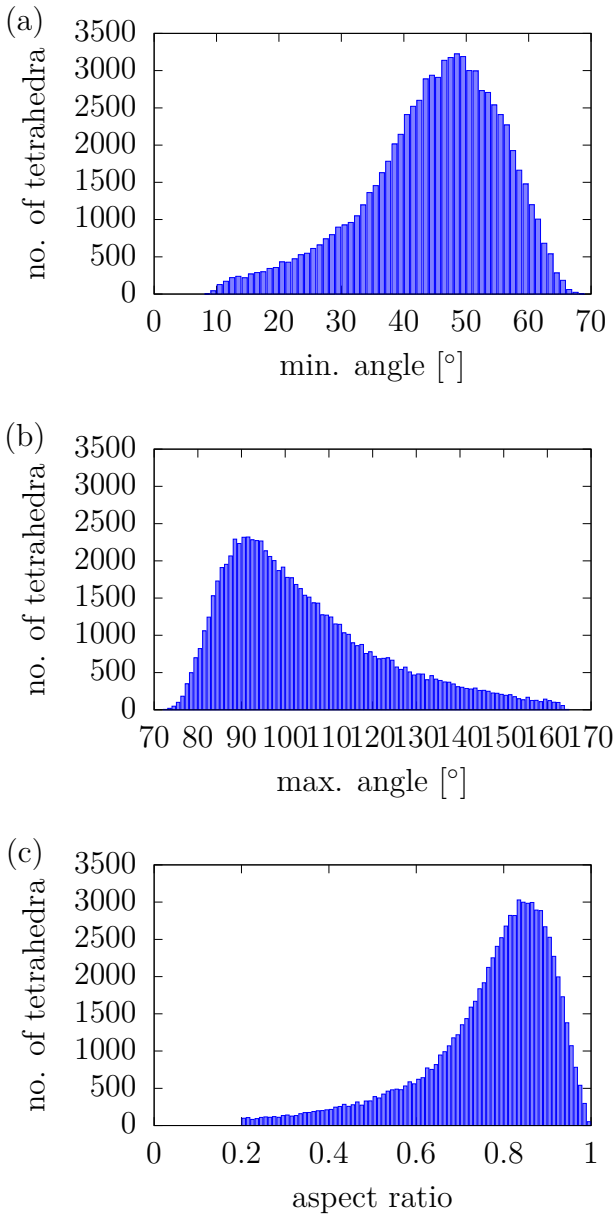


Figure 2.13: Histograms of selected quality measures of the triangulation of variable radii Poisson-disk sampling on DFN and its surrounding matrix. (H=0.01, R=40,A=0.1,F=1). (a): minimal angle ($\geq 8°$), (b): max angle ($\leq 165°$), (c): aspect ratio (≥ 0.2) (In text mentions: p.37)

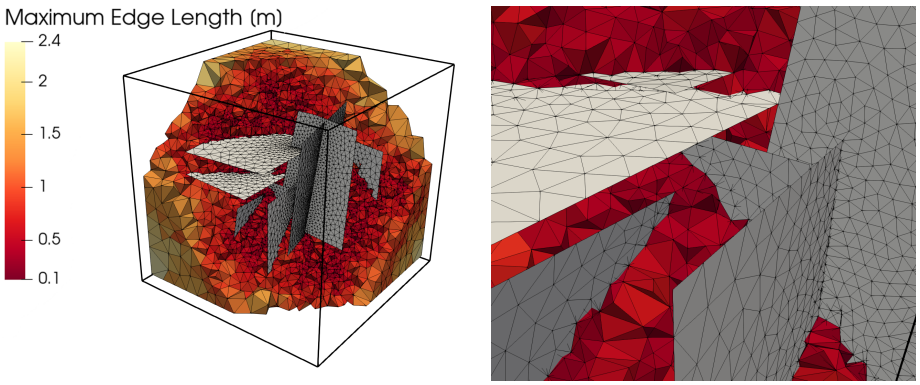


Figure 2.14: (Left) Triangulation of variable radii Poisson-disk sampling of DFN and its surrounding region. (Right) Close up of the conforming mesh. (H=0.25, R=100, A=0.125, F=1). Tetrahedra colored according to their maximal edge length. (In text mentions: pp.37,37)

References

[1] I. Babuska and A. K. Aziz. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, 13(2):214–226, 1976. 11

[2] David A Benson, Tomás Aquino, Diogo Bolster, Nicholas Engdahl, Christopher V Henri, and Daniel Fernandez-Garcia. A comparison of Eulerian and Lagrangian transport and non-linear reaction algorithms. *Advances in Water Resources*, 99:15–37, 2017. 11

[3] Marshall Bern, Paul Chew, David Eppstein, and Jim Ruppert. Dihedral bounds for mesh generation in high dimensions. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '95, pages 189–196, USA, 1995. Society for Industrial and Applied Mathematics. 18

[4] Inga Berre, Florian Doster, and Eirik Keilegavlen. Flow in fractured porous media: A review of conceptual models and discretization approaches. *Transport in Porous Media*, Oct 2018. 9

[5] Stefano Berrone, Sandra Pieraccini, and Stefano Scialo. A PDE-constrained optimization formulation for discrete fracture network flows. *SIAM J. Sci. Comput.*, 35(2):B487–B510, 2013. 9

[6] Stefano Berrone, Stefano Scialò, and Fabio Vicini. Parallel meshing, discretization, and computation of flow in massive discrete fracture networks. *SIAM Journal on Scientific Computing*, 41(4):C317–C338, 2019. 11

[7] Charles Boivin and Carl Ollivier-gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries, 2001. 13

[8] Andrea Borio, Alessio Fumagalli, and Stefano Scialò. Comparison of the response to geometrical complexity of methods for unstationary simulations in discrete fracture networks with conforming, polygonal, and non-matching grids. *Computational Geosciences*, 25(1):143–162, 2021. 11

[9] Robert Bridson. Fast Poisson disk sampling in arbitrary dimensions. In *SIGGRAPH '07*, 2007. xiii, 13, 20, 31, 36

[10] M. C. Cacas, E. Ledoux, G. De Marsily, A. Barbreau, P. Calmels, B. Gaillard, and R. Margritta. Modeling fracture flow with a stochastic discrete fracture network: Calibration and validation: 2. The transport model. *Water Resour. Res.*, 26(3):491–500, 1990. 9

[11] Siu-Wing Cheng, Tamal K. Dey, Herbert Edelsbrunner, Michael A. Facello, and Shang-Hua Teng. Sliver exudation. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry*, SCG '99, pages 1–13, New York, NY, USA, 1999. Association for Computing Machinery. 18, 25

[12] L. Paul Chew. Guaranteed-quality Delaunay meshing in 3d (short version). In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, SCG '97, pages 391–393, New York, NY, USA, 1997. Association for Computing Machinery. 13

[13] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, January 1986. 13

[14] Nicholas Dwork, Corey A. Baron, Ethan M.I. Johnson, Daniel O'Connor, John M. Pauly, and Peder E.Z. Larson. Fast variable density poisson-disc sample generation with directional variation for compressed sensing in mri. *Magnetic Resonance Imaging*, 77:186–193, 2021. xiii, 9, 13, 20, 31, 36

[15] Mohamed S. Ebeida, Andrew A. Davidson, Anjul Patney, Patrick M. Knupp, Scott A. Mitchell, and John D. Owens. Efficient maximal Poisson-disk sampling. *ACM Trans. Graph.*, 30(4), July 2011. 13

[16] Mohamed S. Ebeida, Scott A. Mitchell, Andrew A. Davidson, Anjul Patney, Patrick M. Knupp, and John D. Owens. Efficient and good Delaunay meshes from random points. *Computer-Aided Design*, 43(11):1506–1515, 2011. Solid and Physical Modeling 2011. 13

[17] Mohamed S. Ebeida, Scott A. Mitchell, Anjul Patney, Andrew A. Davidson, and John D. Owens. A simple algorithm for maximal Poisson-disk sampling in high dimensions. *Computer Graphics Forum*, 2012. 13

[18] J Erhel, J-R de Dreuzy, and B Poirriez. Flow simulation in three-dimensional discrete fracture networks. *SIAM J. Sci. Comput.*, 31(4):2688–2705, 2009. 9

[19] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000. 14

[20] Alessio Fumagalli, Eirik Keilegavlen, and Stefano Scialò. Conforming, non-conforming and non-matching discretization couplings in discrete fracture network simulations. *Journal of Computational Physics*, 376:694–712, 2019. 11

[21] HH Gerke and M Th Van Genuchten. A dual-porosity model for simulating the preferential movement of water and solutes in structured porous media. *Water Rescour. Res.*, 29(2):305–319, 1993. 8

[22] Jianwei Guo, Dong-Ming Yan, Li Chen, Xiaopeng Zhang, Oliver Deussen, and Peter Wonka. Tetrahedral meshing via maximal Poisson-disk sampling. *Computer Aided Geometric Design*, 43:186–199, 2016. Geometric Modeling and Processing 2016. 25

[23] Jianwei Guo, Dong-Ming Yan, Xiaohong Jia, and Xiaopeng Zhang. Efficient maximal poisson-disk sampling and remeshing on surfaces. *Computers & Graphics*, 46:72–79, 2015. Shape Modeling International 2014. 13

[24] Jianwei Guo, Dongming Yan, Guanbo Bao, Weiming Dong, Xiaopeng Zhang, and Peter Wonka. Efficient triangulation of poisson-disk sampled point sets. *The Visual Computer*, 30(6-8):773–785, may 2014. KAUST Repository Item: Exported on 2020-10-01 Acknowledgements: This research was partially funded by National Natural Science

Foundation of China (Nos. 61372168, 61172104, 61331018, and 61271431), the KAUST Visual Computing Center, and the National Science Foundation. 13

[25] Teklu Hadgu, Satish Karra, Elena Kalinina, Nataliia Makedonska, Jeffrey D. Hyman, Katherine Klise, Hari S. Viswanathan, and Yifeng Wang. A comparative study of discrete fracture network and equivalent continuum models for simulating flow and transport in the far field of a hypothetical nuclear waste repository in crystalline host rock. *Journal of Hydrology*, 553:59 – 70, 2017. 9, 12

[26] J. D. Hyman, , S. Karra, J. W. Carey, C. W. Gable, H. S. Viswanathan, E. Rougier, and Z. Lei. Discontinuities in effective permeability due to fracture percolation. *Mech. Mater.*, 119:25 – 33, 2018. 12

[27] J. D. Hyman. Flow channeling in fracture networks: Characterizing the effect of density on preferential flow path formation. *Water Resources Research*, 2020. 12

[28] J. D. Hyman, M. Dentz, A. Hagberg, and P. Kang. Emergence of stable laws for first passage times in three-dimensional random fracture networks. *Phys. Rev. Lett.*, 123(24):248501, 2019. 12

[29] J. D. Hyman, M. Dentz, A. Hagberg, and P. Kang. Linking structural and transport properties in three-dimensional fracture networks. *J. Geophys. Res. Sol. Ea.*, 2019. 9

[30] J. D. Hyman, C. W. Gable, S. L. Painter, and N. Makedonska. Conforming Delaunay triangulation of stochastically generated three dimensional discrete fracture networks: A feature rejection algorithm for meshing strategy. *SIAM J. Sci. Comput.*, 36(4):A1871–A1894, 2014. 9, 11, 19

[31] J. D. Hyman and J. Jiménez-Martínez. Dispersion and mixing in three-dimensional discrete fracture networks: Nonlinear interplay between structural and hydraulic heterogeneity. *Water Resources Research*, 54(5):3243–3258, 2018. 9

[32] J. D. Hyman, Joaquin Jimenez-Martinez, Carl W Gable, Philip H Stauffer, and Rajesh J Pawar. Characterizing the impact of fractured caprock heterogeneity on supercritical CO_2 injection. *Transp. Porous Media*, 131(3):935–955, 2020. 12

[33] Jeffrey D Hyman, Satish Karra, Nataliia Makedonska, Carl W Gable, Scott L Painter, and Hari S Viswanathan. dfnworks: A discrete fracture network framework for modeling subsurface flow and transport. *Computers & Geosciences*, 84:10–19, 2015. 11, 32

[34] Jeffrey D Hyman, Satish Karra, Nataliia Makedonska, Carl W Gable, Scott L Painter, and Hari S Viswanathan. dfnworks: A discrete fracture network framework for modeling subsurface flow and transport. *Computers & Geosciences*, 84:10–19, 2015. 25

[35] Jeffrey D. Hyman, Harihar Rajaram, Shriram Srinivasan, Nataliia Makedonska, Satish Karra, Hari Viswanathan, and Gowri Srinivasan. Matrix diffusion in fractured media: New insights into power law scaling of breakthrough curves. *Geophys. Res. Lett.*, 46(23):13785–13795, 2019. 12

[36] Jeffrey D. Hyman, Matthew R. Sweeney, Carl W. Gable, Daniil Svyatsky, Konstantin Lipnikov, and J. David Moulton. Flow and transport in three-dimensional discrete fracture matrix models using mimetic finite differencing on a conforming multi-dimensional mesh. *Journal of Computational Physics*, (Submitted). 25

[37] P. Kang, J. D. Hyman, W. S. Han, and M. Dentz. Anomalous transport in three-dimensional discrete fracture networks: Interplay between aperture heterogeneity and particle injection modes. *Water Resour. Res.*, 2020. 12

[38] S Karra, N Makedonska, HS Viswanathan, SL Painter, and JD Hyman. Effect of advective flow in fractures and matrix diffusion on natural gas production. *Water Resour. Res.*, 51(10):8646–8657, 2015. 12

[39] Ahmed Khamayseh and Andrew Kuprat. Anisotropic smoothing and solution adaption for unstructured grids. *International Journal for Numerical Methods in Engineering*, 39(18):3163–3174, 1996. 12

[40] Johannes Krotz, Matthew R. Sweeney, Carl W. Gable, Jeffrey D. Hyman, and Juan M. Restrepo. Variable resolution poisson-disk sampling for meshing discrete fracture networks. *Journal of Computational and Applied Mathematics*, 407:114094, 2022. 8

[41] P.C. Lichtner, G.E. Hammond, C. Lu, S. Karra, G. Bisht, B. Andre, R.T. Mills, and J. Kumar. PFLOTRAN user manual: A massively parallel reactive flow and transport model for describing surface and subsurface processes. Technical report, (Report No.: LA-UR-15-20403) Los Alamos National Laboratory, 2015. 14

[42] Peter Lichtner and Satish Karra. Modeling multiscale-multiphase-multicomponent reactive flows in porous media: Application to co_2 sequestration and enhanced geothermal energy using PFLOTRAN. In *Al-Khoury, R., Bundschuh, J. (eds.) Computational Models for CO_2 Geo-sequestration & Compressed Air Energy Storage (http://www.crcnetbase.com/doi/pdfplus/10)*, pages 81–136. CRC Press, 2014. 8

[43] Yehong Liu and Guosheng Yin. The Delaunay triangulation learner and its ensembles. *Computational Statistics & Data Analysis*, page 107030, 2020. 14

[44] JCS Long, JS Remer, CR Wilson, and PA Witherspoon. Porous media equivalents for networks of discontinuous fractures. *Water Resour. Res*, 18(3):645–658, 1982. 9

[45] A. E. Lovell, S. Srinivasan, S. Karra, D. O'Malley, N. Makedonska, H. S. . Viswanathan, G. Srinivasan, J. W. Carey, and L. P. Frash. Extracting hydrocarbon from shale: An investigation of the factors that influence the decline and the tail of the production curve. *Water Resour. Res.*, 2018. 12

[46] N. Makedonska, J. D. D Hyman, S. Karra, S. L Painter, C. W. W Gable, and H. S Viswanathan. Evaluating the effect of internal aperture variability on transport in kilometer scale discrete fracture networks. *Adv. Water Resour.*, 94:486–497, 2016. 12

[47] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, and Noel Walkington. A Delaunay based numerical method for three dimensions: Generation, formulation, and partition. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 683–692, New York, NY, USA, 1995. Association for Computing Machinery. 13

[48] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, Noel Walkington, and Han Wang. Control volume meshes using sphere packing: Generation, refinement and coarsening. In *In Fifth International Meshing Roundtable*, pages 47–61, 1996. 13

[49] Scott Mitchell, Alexander Rand, Mohamed Ebeida, and Chandrajit Bajaj. Variable radii Poisson-disk sampling. *Proceedings of the 24th Canadian Conference on Computational Geometry, CCCG 2012*, 01 2012. 9, 13, 14

[50] M. K. Mudunuru, S. Karra, N. Makedonska, and T. Chen. Sequential geophysical and flow inversion to characterize fracture networks in subsurface systems. *Stat. Anal. Data. Min.*, 10(5):326–342, 2017. 12

[51] Michael Murphy, David M Mount, and Carl W Gable. A point-placement strategy for conforming Delaunay tetrahedralization. *International Journal of Computational Geometry & Applications*, 11(06):669–682, 2001. 9, 10, 21

[52] H. Mustapha and K. Mustapha. A new approach to simulating flow in discrete fracture networks with an optimized mesh. *SIAM J. Sci. Comput.*, 29:1439, 2007. 9, 11

[53] Hussein Mustapha, Roussos Dimitrakopoulos, Thomas Graf, and Abbas Firoozabadi. An efficient method for discretizing 3d fractured media for subsurface flow and transport simulations. *International Journal for Numerical Methods in Fluids*, 67(5):651–670, 2011. 9, 11

[54] Engineering National Academies of Sciences, Medicine, et al. *Characterization, modeling, monitoring, and remediation of fractured rock*. National Academies Press, 2020. 10

[55] National Research Council. *Rock fractures and fluid flow: contemporary understanding and applications*. National Academy Press, 1996. 10

[56] Shlomo P Neuman and Joseph S Depner. Use of variable-scale pressure test data to estimate the log hydraulic conductivity covariance and dispersivity of fractured granites near oracle, arizona. *J. Hydrol.*, 102(1-4):475–501, 1988. 8

[57] S.P. Neuman. Trends, prospects and challenges in quantifying flow and transport through fractured rocks. *Hydrogeol. J.*, 13(1):124–147, 2005. 8, 10

[58] A. Wille Nordqvist, Y. W. Tsang, C. F. Tsang, Björn Dverstorp, and Johan Andersson. A variable aperture fracture network model for flow and transport in fractured rocks. *Water Resources Research*, 28(6):1703–1713, 1992. 9

[59] G Pichot, J Erhel, and J-R de Dreuzy. A mixed hybrid mortar method for solving flow in discrete fracture networks. *Appl. Anal.*, 89(10):1629–1643, 2010. 9

[60] G Pichot, J Erhel, and J-R de Dreuzy. A generalized mixed hybrid mortar method for solving flow in stochastic discrete fracture networks. *SIAM J. Sci. Comput.*, 34(1):B86–B105, 2012. 9

[61] Karsten Pruess, Curtis M Oldenburg, and GJ Moridis. Tough2 user's guide version 2. 1999. 14

[62] M Cecilia Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *International journal for numerical methods in Engineering*, 20(4):745–756, 1984. 12

[63] Maria-Cecilia Rivara. Mesh refinement processes based on the generalized bisection of simplices. *SIAM J. Numer. Anal.*, 21(3):604–613, 1984. 12

[64] T. Sherman, J. D. Hyman, M. Dentz, and D. Bolster. Characterizing the influence of fracture density on network scale transport. *J. Geophys. Res. Sol. Ea.*, 125(1):e2019JB018547, 2020. e2019JB018547 10.1029/2019JB018547. 12

[65] Gilbert Strang, George J. Fix, and D. S. Griffin. An Analysis of the Finite-Element Method. *Journal of Applied Mechanics*, 41(1):62–62, 03 1974. 11

[66] Dafna Talmor, Guy Blelloch, Alan M. Frieze, Noel J. Walkington, and Shang hua Teng. Well-spaced points for numerical methods. Technical report, 1997. 13

[67] Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation.

In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, New York, NY, USA, 2009. Association for Computing Machinery. 25

[68] YW Tsang, CF Tsang, FV Hale, and B Dverstorp. Tracer transport in a stochastic continuum model of fractured media. *Water Resour. Res*, 32(10):3077–3092, 1996. 8

[69] Dong Ming Yan, Jian Wei Guo, Bin Wang, Xiao Peng Zhang, and Peter Wonka. A survey of blue-noise sampling and its applications. *Journal of Computer Science and Technology*, 30(3):439–452, may 2015. Publisher Copyright: © 2015, Springer Science+Business Media New York. 13

[70] Robert W Zimmerman, Gang Chen, Teklu Hadgu, and Gudmundur S Bodvarsson. A numerical dual-porosity model with semianalytical treatment of fracture/matrix flow. *Water Resour. Res*, 29(7):2127–2137, 1993. 8

[71] M. ZLAMAL. On the finite element method. *Numerische Mathematik*, 12:394–409, 1968. 11, 14

[72] G Zyvoloski. FEHM: A control volume finite element code for simulating subsurface multi-phase multi-fluid heat and mass transfer. *Los Alamos Unclassified Report LA-UR-07-3359*, 2007. 14

Chapter 3

A Hybrid Monte Carlo, Discontinuous Galerkin method for linear kinetic transport equations

3.1 Disclosure

This chapter is, up to formatting identical to the preprint of the same name [23]. The manuscript is collaborative work with Cory D. Hauck and Ryan G. McClarren. At the time this disseration is written the article has been submitted to the Journal of computational Physics and is under review. A preprint is available under https://doi.org/10.48550/arXiv.2312.04217. The ideas and results presented build on previous work by Cory D. Hauck and Ryan G. McClarren. Ideas and theoretical results newly presented in this paper were worked out by Johannes Krotz under their guidance. Numerical experiments presented were executed by Johannes Krotz using software implemented in Matlab also by Johannes Krotz. All three cowrote and edited the paper together.

3.2 Abstract

We present a hybrid method for time-dependent particle transport problems that combines Monte Carlo (MC) estimation with deterministic solutions based on discrete ordinates. For spatial discretizations, the MC algorithm computes a piecewise constant solution and the discrete ordinates uses bilinear discontinuous finite elements. From the hybridization of the problem, the resulting problem solved by Monte Carlo is scattering free, resulting in a simple, efficient solution procedure. Between time steps, we use a projection approach to "relabel" collided particles as uncollided particles. From a series of standard 2-D Cartesian test problems we observe that our hybrid method has improved accuracy and reduction in computational complexity of approximately an order of magnitude relative to standard discrete ordinates solutions.

3.3 Introduction

Numerical methods for kinetic transport equations are commonly divided into two classes: deterministic and Monte Carlo. Each of these approaches has strengths and weaknesses that complement the other.

Deterministic methods [25] directly discretize phase space (physical space, direction of flight, particle energy) as well as time, in the time-dependent setting. For this large seven-dimensional space (three for physical space, two for direction of flight, one for energy, and one for time), it is difficult to construct high resolution solutions for general problems. Indeed, the number of operations and the memory footprint required for deterministic solvers can be a challenge, even for leadership-class computers.

Monte Carlo methods [35], on the other hand, use sampling techniques to simulate particle transport processes. In its most basic form, the Monte-Carlo procedure is a computational analog of the actual physical processes being simulated: particles are sampled from sources and boundary conditions, then tracked as they stream through the domain, and along the way undergo scattering or absorption interactions with the material medium. As the particle traverses the physical domain, it contributes to integrated quantities of interest such as particle density or net fluence through a surface. For linear problems, the central limit theorem implies that the Monte Carlo solution is exact in the limit of an infinite number of samples [35]. Unlike deterministic methods, Monte Carlo methods are easy to extend to complicated 3-D geometries and can handle physical processes (such as particle interactions with the background material) in a continuous manner. Nevertheless, the uncertainty in Monte Carlo methods, as expressed in the standard deviation of an estimate, scales like $N^{-1/2}$, where N is the number of sample particles. Additionally, Monte Carlo methods are not well-suited for obtaining uniform spatial estimates due to the difficulty of getting sufficient samples in every region of the physical domain. Moreover, for nonlinear problems such as thermal radiative transfer, the Monte Carlo approach loses some of its attractive properties. For example the discretization of material temperature, to which the particles are coupled, means that an exact solution is not obtained in the limit of infinite samples [44, 12, 27]. Nevertheless, producing efficient, accurate Monte Carlo calculations is an active area of research [34, 36].

Hybrid methods have been developed to harmonize the benefits of Monte Carlo and deterministic methods while minimizing their respective drawbacks. For steady-state nuclear reactor problems, methods such as COMET [28, 45] use local Monte Carlo calculations to estimate properties of solutions in macroscopic regions of the problem and then use a

deterministic procedure to couple these regions together. Other work has considered weight windows and other biasing techniques [38, 6, 7, 39, 29] wherein deterministic solutions are used to modify the flight of particles in Monte Carlo calculations so that computational effort is spent more efficiently. High-order low-order (HOLO) schemes [30, 43, 31] have been developed in which Monte Carlo is used to compute a closure term for a low-order, moment-based deterministic calculation.

This work presents a deterministic-Monte Carlo hybrid method for time-dependent problems based on the physics of particle transport. Previous work [21, 9, 10, 22, 41] has shown that treating particles from the beginning of a step to their first collision with a high-resolution discretization in angle, and treating the particles after they scatter with a low-resolution method can give efficient and accurate numerical calculations. Because the scattering process relaxes particles towards a weakly anisotropic angular distribution, one can combine methods that are appropriate for particle streaming for the uncollided particles during a time step with methods that are suitable for weakly anisotropic angular distributions. In previous work, deterministic methods with a large number of angular degrees of freedom were used for the uncollided particles while low-resolution deterministic methods were used for the collided particles. Moreover, it has been shown [41] that the benefits of this splitting approach extend to multigroup problems by applying a coarser energy and angle discretization to the collided particles.

Despite the benefits of deterministic hybrid methods, solutions still require a large number of degrees of freedom for problems with large streaming paths. A natural strategy to address this challenge, which for steady-state problems was first proposed in [3], is to use Monte Carlo for the uncollided particles. Indeed, in many respects this is the ideal situation for a Monte Carlo approach. During a time step, particles are tracked through the computational domain and a non-analog estimator of the solution known as implicit capture is employed, thereby avoiding the need to consider collisions at all. Thus the calculation of the contribution to the solution from uncollided particles is essentially a ray tracing algorithm, which has many efficient implementations on modern computing hardware [2].

The hybrid method considered here uses Monte Carlo to compute the contribution to the solution from uncollided particles and an efficient deterministic calculation for the collided

particles. A key advancement for extending the original steady-state formulation to time-dependent problems is a remapping step that resamples particles from the deterministic collided solution back into the uncollided component. This procedure is critical since, otherwise, the number of uncollided particles will decay exponentially and the hybrid solution will relax to a low-resolution, deterministic approximation of the collided solution [21]. We find that the hybrid approach leads to more accurate solutions obtained using lower computational complexity than pure deterministic calculations. An additional benefit of the hybrid is that, away from domain boundaries, it reduces to the uncollided discretization, which unlike standard Monte-Carlo methods, captures the diffusion limit in optically thick regimes [21].

The remainder of the paper is organized as follows. In Section 2, we introduce the hybrid method in the context of a single-group transport equation that is independent of particle energy. We also summarize the numerical methods used for the uncollided and collided components of the hybrid. In Section 3, we present numerical results for several standard test problems in a reduced two-dimensional geometry in physical space. In Section 4, we summarize findings and present directions for future work. A short appendix describes the Monte Carlo implementation of a boundary for one of the test problems.

3.4 Basics of the Hybrid Method

3.4.1 Transport equation

Let $X \subset \mathbb{R}^3$ be a spatial domain with Lipschitz boundary and let $\mathbb{S}^2$ be the unit sphere in $\mathbb{R}^3$. Let $\Psi = \Psi(\boldsymbol{x}, \boldsymbol{\Omega}, t)$ be the angular flux depending on the position $\boldsymbol{x} = (x, y, z) \in X$, the direction of flight $\boldsymbol{\Omega} \in \mathbb{S}^2$ and time $t > 0$. We assume that Ψ is governed by the linear transport equation

$$\frac{1}{c}\partial_t \Psi + \Omega \cdot \nabla_{\boldsymbol{x}} \Psi + \sigma_{\mathrm{t}} \Psi = \frac{\sigma_{\mathrm{s}}}{4\pi} \langle \Psi \rangle + Q, \qquad \boldsymbol{x} \in X, \quad \boldsymbol{\Omega} \in \mathbb{S}^2, \quad t > 0, \tag{3.1}$$

where $\sigma_{\mathrm{t}} = \sigma_{\mathrm{t}}(\boldsymbol{x})$, $\sigma_{\mathrm{s}} = \sigma_{\mathrm{s}}(\boldsymbol{x})$ and $\sigma_{\mathrm{a}} = \sigma_{\mathrm{t}} - \sigma_{\mathrm{s}}$ are the total, scattering, and absorption cross-sections of the material, respectively; $Q = Q(\boldsymbol{x}, \boldsymbol{\Omega}, t)$ is a known particle source; and

angle brackets denote integration over the unit sphere:

$$\langle \Psi \rangle = \int_{\mathbb{S}^2} \Psi \, d\boldsymbol{\Omega}. \tag{3.2}$$

The constant $c > 0$ is the particle speed; we assume that $c = 1$ for the remainder of this work. The transport equation (3.1) is equipped with initial conditions

$$\Psi(\boldsymbol{x}, \boldsymbol{\Omega}, 0) = \Psi_0(\boldsymbol{x}, \boldsymbol{\Omega}), \quad \boldsymbol{x} \in X, \quad \boldsymbol{\Omega} \in \mathbb{S}^2, \tag{3.3}$$

and boundary condition

$$\Psi(\boldsymbol{x}, \boldsymbol{\Omega}, t) = G(\boldsymbol{x}, \boldsymbol{\Omega}, t) \qquad \boldsymbol{x} \in \partial X, \quad \boldsymbol{\Omega} \cdot \boldsymbol{n}(\boldsymbol{x}) < 0, \tag{3.4}$$

where Ψ_0 and G are known and $\boldsymbol{n}(\boldsymbol{x})$ is the unit outward normal at $\boldsymbol{x} \in \partial X$.

3.4.2 The hybrid method

The hybrid method is based on a first collision source splitting [3]. Let $\Psi = \Psi_\mathrm{u} + \Psi_\mathrm{c}$, where the *uncollided flux* Ψ_u and the *collided flux* Ψ_c satisfy the following system of equations

$$\partial_t \Psi_\mathrm{u} + \Omega \cdot \nabla_{\boldsymbol{x}} \Psi_\mathrm{u} + \sigma_\mathrm{t} \Psi_\mathrm{u} = Q, \tag{3.5a}$$

$$\partial_t \Psi_\mathrm{c} + \Omega \cdot \nabla_{\boldsymbol{x}} \Psi_\mathrm{c} + \sigma_\mathrm{t} \Psi_\mathrm{c} = \frac{\sigma_\mathrm{s}}{4\pi} \left(\langle \Psi_\mathrm{u} \rangle + \langle \Psi_\mathrm{c} \rangle \right). \tag{3.5b}$$

Due to the linearity of (3.1), the splitting in (3.5) is exact; that is, if Ψ_u and Ψ_c solve (3.5a) and (3.5b), respectively, then $\Psi_\mathrm{u} + \Psi_\mathrm{c}$ solves (3.1). In practice, however, (3.5a) and (3.5b) are solved at different resolutions or even with different methods. Typically (3.5a) is solved with a method that has high resolution in angle, and because (3.5a) has no coupling in angle, it is easier to solve than (3.1) and also easy to solve in parallel. Meanwhile (3.5b) inherits the angular coupling in (3.1), but typically requires less angular resolution.

Since (3.5a) has no scattering source, particle densities will be transferred into the collided flux at an exponential rate, thus making the accuracy at which (3.5b) is solved the driving

factor in the overall accuracy. This effect can be mitigated by abusing the autonomous nature of the equations and periodically relabelling the collided flux as uncollided at every time step.

To describe the implementation of the hybrid in more detail, let $\mathcal{T}$ be an operator such that

$$u[t] = \mathcal{T}(t, t', s, \upsilon, b, \lambda_{\mathrm{t}}, \lambda_{\mathrm{s}}) \tag{3.6}$$

where $u[t](\boldsymbol{x}, \boldsymbol{\Omega}) := u(\boldsymbol{x}, \boldsymbol{\Omega}, t)$, satisfies

$$\begin{cases} \partial_t u + \boldsymbol{\Omega} \cdot \nabla_{\boldsymbol{x}} u + \lambda_{\mathrm{t}} u = \dfrac{\lambda_{\mathrm{s}}}{4\pi} \langle u \rangle + s, & \boldsymbol{x} \in X, \quad \boldsymbol{\Omega} \in \mathbb{S}^2, \quad t > t', & \text{(3.7a)} \\ u(\boldsymbol{x}, \boldsymbol{\Omega}, t') = \upsilon(\boldsymbol{x}, \boldsymbol{\Omega}) & \boldsymbol{x} \in X, \quad \boldsymbol{\Omega} \in \mathbb{S}^2, & \text{(3.7b)} \\ u(\boldsymbol{x}, \boldsymbol{\Omega}, t) = b(\boldsymbol{x}, \boldsymbol{\Omega}, t) & \boldsymbol{x} \in \partial X, \quad \boldsymbol{\Omega} \cdot \boldsymbol{n}(\boldsymbol{x}) < 0, \quad t > t'. & \text{(3.7c)} \end{cases}$$

with source $s = s(\boldsymbol{x}, \boldsymbol{\Omega}, t)$. Using the operator $\mathcal{T}$, we can write

$$\Psi[t_{n+1}] = \mathcal{T}(t_{n+1}, t_n, Q, \Psi[t_n], G, \sigma_{\mathrm{t}}, \sigma_{\mathrm{s}}), \tag{3.8}$$

$$\Psi_{\mathrm{u}}[t_{n+1}] = \mathcal{T}(t_{n+1}, t_n, Q_{\mathrm{u}}, \Psi[t_n], G, \sigma_{\mathrm{t}}, 0), \qquad Q_{\mathrm{u}} = Q \tag{3.9}$$

$$\Psi_{\mathrm{c}}[t_{n+1}] = \mathcal{T}(t_{n+1}, t_n, Q_{\mathrm{c}}, 0, 0, \sigma_{\mathrm{t}}, \sigma_{\mathrm{s}}), \qquad Q_{\mathrm{c}} = \frac{\sigma_{\mathrm{s}}}{4\pi} \langle \Psi_{\mathrm{u}} \rangle. \tag{3.10}$$

We simulate the system (3.5) using a Monte Carlo method for the uncollided equation (3.5a) and a deterministic discretization of the collided equation (3.5b). Let

$$\mathcal{T}_{\mathrm{MC}}(t, t', s, \upsilon_{\mathrm{MC}}, b, \lambda_{\mathrm{t}}, \lambda_{\mathrm{s}}; N_p) \tag{3.11}$$

be the Monte Carlo approximation to (3.7) given a particle representation υ_{MC} of υ and using N_p pseudo-particles to represent the distribution of particles in phase space introduced by the source s over the internal (t', t). Similarly,

$$\mathcal{T}_{\mathrm{SN}}(t, t', s, v, b, \lambda_{\mathrm{t}}, \lambda_{\mathrm{s}}; N, N_{\boldsymbol{x}}) \tag{3.12}$$

denote the S_N-DG approximation of (3.7) using a level N set of ordinates, $N_{\boldsymbol{x}}$ spatial cells per dimension with $\mathbb{Q}_1$ elements, and a backward Euler time discretization to get from t' to t.

(The Monte Carlo method and S_N-DG method are presented in more detail below.) Then given N_p, N, and $N_{\boldsymbol{x}}$, and a Monte Carlo approximation ψ^n of $\Psi(t_n)$, let

$$\psi_{\mathrm{u}}^{n+1} = \mathcal{T}_{\mathrm{MC}}(t_{n+1}, t_n, Q_{\mathrm{u}}, \psi^n, G, \sigma_{\mathrm{t}}, 0; N_p), \qquad Q_{\mathrm{u}} = Q \tag{3.13}$$

$$\psi_{\mathrm{c}}^{n+1} = \mathcal{T}_{\mathrm{SN}}(t_{n+1}, t_n, Q_{\mathrm{c}}, 0, 0, \sigma_{\mathrm{t}}, \sigma_{\mathrm{s}}; N, N_{\boldsymbol{x}}), \qquad Q_{\mathrm{c}} = \frac{\sigma_{\mathrm{s}}}{4\pi}\langle\psi_{\mathrm{u}}\rangle_{\mathrm{MC}} \tag{3.14}$$

$$\mathcal{R}\psi_{\mathrm{c}}^{n+1} = \mathcal{T}_{\mathrm{MC}}(t_{n+1}, t_n, Q_{\mathrm{r}}, 0, 0, \sigma_{\mathrm{t}}, 0; N_p), \qquad Q_{\mathrm{r}} = Q_{\mathrm{c}} + \frac{\sigma_{\mathrm{s}}}{4\pi}\langle\psi_{\mathrm{c}}\rangle_{\mathrm{SN}} \tag{3.15}$$

$$\psi^{n+1} = \psi_{\mathrm{u}}^{n+1} + \mathcal{R}\psi_{\mathrm{c}}^{n+1} \tag{3.16}$$

where $\mathcal{R}$ is the relabeling operator and $\langle\cdot\rangle_{\mathrm{MC}}$ and $\langle\cdot\rangle_{\mathrm{SN}}$ denote approximation of the angular integral over $\mathbb{S}^2$ with the respective method. When equations (3.13) through (3.15) are solved sequentially, ψ_{u}^{n+1} is a Monte Carlo approximation of (3.5a) with initial condition ψ^n and $\mathcal{R}\psi_{\mathrm{c}}^{n+1}$ is a Monte Carlo approximation of (3.5b) with zero initial condition. Thus ψ^{n+1} is a Monte Carlo approximation of (3.1) with initial condition ψ^n.

3.4.3 Discrete ordinate-discontinuous Galerkin

The discrete ordinates (S_N) method [5] approximates (3.7) by replacing the angular integral $\langle u\rangle$ by a discrete quadrature and then solving the resulting equation for the angles in the quadrature. This procedure yields a system of equations that depend only on space and time and can be further discretized by a variety of methods. Let

$$\{\boldsymbol{\Omega}_q\}_{q=1}^{N_{\boldsymbol{\Omega}}} \quad \text{and} \quad \{\omega_q\}_{q=1}^{N_{\boldsymbol{\Omega}}} \tag{3.17}$$

be the angles and associated weights of the S_N quadrature, where $N_{\boldsymbol{\Omega}} = N_{\boldsymbol{\Omega}}(N)$ depends on the specific type of quadrature set being used. After discretizing in angle and applying an implicit Euler time discretization, the following semi-discrete system is obtained for each

$q \in \{1, ..., N_{\mathbf{\Omega}}\}$ and $n \in \{0, 1, 2, \dots\}$,

$$\begin{cases} \dfrac{1}{\Delta t}\left(u_q^{n+1} - u_q^n\right) + \mathbf{\Omega}_q \cdot \nabla_{\boldsymbol{x}} u_q^{n+1} + \lambda_{\mathrm{t}} u_q^{n+1} = \dfrac{\lambda_{\mathrm{s}}}{4\pi} \displaystyle\sum_{r=1}^{N_{\mathbf{\Omega}}} \omega_r u_r^{n+1} + s_q^{n+1}, & \boldsymbol{x} \in X, \quad (3.18a) \\ u_q^{n+1}(\boldsymbol{x}) = b_q^{n+1}(\boldsymbol{x}), & \boldsymbol{x} \in \partial X_q^-, \quad (3.18b) \end{cases}$$

where $\partial X_q^- = \{\boldsymbol{x} \in X : \mathbf{\Omega}_q \cdot \boldsymbol{n}(\boldsymbol{x}) < 0\}$, $b_q^n(\boldsymbol{x}) = b(\boldsymbol{x}, \mathbf{\Omega}_q, t_n)$, $s_q^n(\boldsymbol{x}) = s(\boldsymbol{x}, \mathbf{\Omega}_q, t_n)$, and $u_q^n(\boldsymbol{x}) \approx u(\boldsymbol{x}, \mathbf{\Omega}_q, t_n)$ is the approximation on the temporal and angular grid. After reassigning

$$u_q \leftarrow u_q^{n+1}, \qquad s_q \leftarrow s_q^{n+1} + u_q^n, \qquad \lambda_{\mathrm{t}} \leftarrow \lambda_{\mathrm{t}} + \frac{1}{\Delta t}, \quad \text{and} \quad b_q \leftarrow b_q^{n+1}, \qquad (3.19)$$

the discretization in (3.18a) can be written in the equivalent steady-state form

$$\begin{cases} \mathbf{\Omega}_q \cdot \nabla_{\boldsymbol{x}} u_q + \lambda_{\mathrm{t}} u_q = \lambda_{\mathrm{s}} \bar{\boldsymbol{u}} + s_q, & \boldsymbol{x} \in X \quad (3.20a) \\ u_q(\boldsymbol{x}) = b_q(\boldsymbol{x}), & \boldsymbol{x} \in \partial X_q^- \quad (3.20b) \end{cases}$$

where $\boldsymbol{u} = (u_1, \dots, u_{N_{\mathbf{\Omega}}})^\top$, $\bar{\boldsymbol{u}} := \frac{1}{4\pi} \sum_r \omega_r u_r$.

We discretize (3.20a) in physical space with a discontinuous Galerkin method and upwind numerical fluxes. The method by now is fairly standard (see for example [9, 20]) and is often used because of its accuracy in scattering-dominated regimes relative to upwind finite-difference and finite-volume methods [1, 24, 33, 19]. Because the DG method is well-known, we summarize it only briefly for the case of a two-dimensional Cartesian mesh with rectangular cells, which is sufficient for all of the numerical tests in Section 3.5. Let X be divided into open sets $C_{i,j}$ that are squares with side lengths h centered at (x_i, y_j), and let $V_h = \{v \in L^2(X) \ : v|_{C_{ij}} \in \mathbb{Q}_1\}$. The goal will be to find an approximation of the weak solution of equation (3.20a); that is, find $\boldsymbol{u}^h = (u_1^h, \dots, u_{N_{\mathbf{\Omega}}}^h)^\top \in [V_h]^{N_{\mathbf{\Omega}}} := V_h \underbrace{\times \cdots \times}_{N_{\mathbf{\Omega}} \text{ times}} V_h$ such that

$$\mathcal{A}_q^{(i,j)}(u_q^h, v_q^h) + \mathcal{P}_q^{(i,j)}(u_q^h, v_q^h) = \mathcal{M}_q^{(i,j)}(u_q^h, v_q^h) + \mathcal{R}^{(i,j)}(\boldsymbol{u}^h, v_q^h) + \mathcal{S}_q^{(i,j)}(v_q^h) + \mathcal{B}_q^{(i,j)}(v_q^h) \quad (3.21)$$

for all $i,j \in \{1,\dots,N_{\boldsymbol{x}}\}$, $q \in \{1,...,N\}$, and $\boldsymbol{v}^h \in V_h^{N_{\boldsymbol{\Omega}}}$. Here

$$\mathcal{A}_q^{(i,j)}(u_q^h, v_q^h) = -\int_{C_{i,j}} (\boldsymbol{\Omega}_q \cdot \nabla_{\boldsymbol{x}} v_q^h) u_q^h d\boldsymbol{x} + \lambda_{\mathrm{t}} \int_{C_{i,j}} v_q^h u_q^h d\boldsymbol{x}, \tag{3.22}$$

$$\mathcal{P}_q^{(i,j)}(u_q^h, v_q^h) = \int_{(\partial C_{i,j})_q^+} (\boldsymbol{\Omega}_q \cdot \boldsymbol{n}) v_q^{h,-} v_q^{h,-} ds(\boldsymbol{x}), \qquad \mathcal{M}_q^{(i,j)}(u_q^h, v_q^h) = \int_{(\partial C_{i,j})_q^-} (\boldsymbol{\Omega}_q \cdot \boldsymbol{n}) v_q^{h,-} u_q^{h,+} ds(\boldsymbol{x}), \tag{3.23}$$

$$\mathcal{R}^{(i,j)}(\boldsymbol{u}^h, v_q^h) = \lambda_{\mathrm{s}} \int_{C_{i,j}} \bar{\boldsymbol{u}}^h v_q^h d\boldsymbol{x}, \quad \mathcal{S}_q^{(i,j)}(v_q^h) = \int_{C_{i,j}} s_q v_q^h d\boldsymbol{x}, \quad \mathcal{B}_q^{(i,j)}(v_q^h) = \int_{C_{i,j} \cap \partial X_q^-} b_q v_q^h ds(\boldsymbol{x}), \tag{3.24}$$

$$v_q^{h,\pm}(\boldsymbol{x}) = \lim_{\vartheta \to 0^+} v_q^h(\boldsymbol{x} \pm \vartheta \boldsymbol{n}), \qquad \text{and} \qquad (\partial C_{i,j})_q^{\pm} = \{\boldsymbol{x} \in \partial C_{i,j} : \pm \boldsymbol{\Omega}_q \cdot \boldsymbol{n}(\boldsymbol{x}) > 0\} \tag{3.25}$$

We construct $\boldsymbol{u}^h = (u_1^h, \dots, u_N^h)^\top$ as follows: For each $i,j \in \{1,\dots,N_{\boldsymbol{x}}\}$ and $q \in \{1,...,N_{\boldsymbol{\Omega}}\}$, let

$$u_q^h(\boldsymbol{x}) = \sum_{|\boldsymbol{k}|_\infty \le 1} \alpha_{q,\boldsymbol{k}}^{(i,j)} \phi_{\boldsymbol{k}}^{(i,j)}(x,y), \qquad \boldsymbol{x} \in C_{i,j}, \tag{3.26}$$

where $\boldsymbol{k} = (k_1, k_2)$,

$$\phi_{\boldsymbol{k}}^{(i,j)} = P_{k_1}\left(\frac{x - x_i}{h/2}\right) P_{k_2}\left(\frac{y - y_j}{h/2}\right), \tag{3.27}$$

and P_k is the usual Legendre polynomial of degree k on $\xi \in [-1,1]$ with normalization $\int_{-1}^1 P_{k_1}(\xi) P_{k_2}(\xi) d\xi = \frac{2}{2k_1+1}\delta_{k_1,k_2}$. Using this representation for $\boldsymbol{u}^h$, we derive the following linear system for the coefficients $\alpha_{q,\boldsymbol{k}}^{(i,j)}$ from equation (3.21): For each $\boldsymbol{l}$ such that $|\boldsymbol{l}|_\infty \le 1$,

$$\sum_{|\boldsymbol{k}|_\infty \le 1} \left(\boldsymbol{A}_{q,\boldsymbol{l},\boldsymbol{k}}^{(i,j)} + \boldsymbol{P}_{q,\boldsymbol{l},\boldsymbol{k}}^{(i,j)}\right) \alpha_{q,\boldsymbol{k}}^{(i,j)} = \sum_{|\boldsymbol{k}|_\infty \le 1} \left(\boldsymbol{M}_{q,\boldsymbol{l},\boldsymbol{k}}^{(i,j)} \boldsymbol{\alpha}_{q,\boldsymbol{k}}^{(i^*,j^*q)} + \boldsymbol{R}_{\boldsymbol{l},\boldsymbol{k}}^{(i,j)} \bar{\boldsymbol{\alpha}}_{\boldsymbol{k}}^{(i,j)}\right) + \boldsymbol{S}_{q,\boldsymbol{l}}^{(i,j)} + \boldsymbol{B}_{q,\boldsymbol{l}}^{(i,j)}, \tag{3.28}$$

where $\bar{\boldsymbol{\alpha}}_{\boldsymbol{k}}^{(i,j)} = \sum_{q=1}^{N_{\boldsymbol{\Omega}}} w_q \alpha_{q,\boldsymbol{k}}^{(i,j)}$,

$$\boldsymbol{A}_{q,\boldsymbol{l},\boldsymbol{k}}^{(i,j)} = \mathcal{A}_q^{(i,j)}(\phi_{\boldsymbol{k}}^{(i,j)}, \phi_{\boldsymbol{l}}^{(i,j)}), \qquad \boldsymbol{P}_{q,\boldsymbol{l},\boldsymbol{k}}^{(i,j)} = \mathcal{P}_q^{(i,j)}(\phi_{\boldsymbol{k}}^{(i,j)}, \phi_{\boldsymbol{l}}^{(i,j)}) \qquad \boldsymbol{M}_{q,\boldsymbol{l},\boldsymbol{k}}^{(i,j)} = \mathcal{M}_q^{(i,j)}(\phi_{\boldsymbol{k}}^{(i^*,j^*)}, \phi_{\boldsymbol{l}}^{(i,j)}) \tag{3.29}$$

$$\boldsymbol{B}_{q,\boldsymbol{l}}^{(i,j)} = \mathcal{B}_q^{(i,j)}(\phi_{\boldsymbol{l}}^{(i,j)}), \qquad \boldsymbol{S}_{q,\boldsymbol{l}}^{(i,j)} = \mathcal{S}_q^{(i,j)}(\phi_{\boldsymbol{l}}^{(i,j)}), \qquad \boldsymbol{R}_{\boldsymbol{l},\boldsymbol{k}}^{(i,j)} = \mathcal{R}^{(i,j)}(\phi_{\boldsymbol{k}}^{(i,j)}, \phi_{\boldsymbol{l}}^{(i,j)}) \tag{3.30}$$

and, given the components $\boldsymbol{n} = (n_x, n_y)$ of the outward normal, the indices i^*, j^* are given by

$$i^*(\boldsymbol{x}) = i + n_x(\boldsymbol{x}) \quad \text{and} \quad j^*(\boldsymbol{x}) = j + n_y(\boldsymbol{x}). \tag{3.31}$$

To improve readability, we rewrite equation (3.28) as a matrix equation with respect to the indices $\boldsymbol{k}$ and $\boldsymbol{l}$:

$$\left(\boldsymbol{A}_q^{(i,j)} + \boldsymbol{P}_q^{(i,j)}\right) \boldsymbol{\alpha}_q^{(i,j)} = \boldsymbol{R}^{(i,j)} \bar{\boldsymbol{\alpha}}^{(i,j)} + \boldsymbol{M}_q^{(i,j)} \boldsymbol{\alpha}_q^{(i^*,j^*)} + \boldsymbol{B}_q^{(i,j)} + \boldsymbol{S}_q^{(i,j)}. \tag{3.32}$$

The organization of the operators in (3.32) reflects a standard solution strategy combining source iteration and sweeping. In source iteration, $\bar{\boldsymbol{\alpha}}^{(i,j)}$ is lagged; that is, given an iteration index ℓ:

$$\boldsymbol{\alpha}_q^{(i,j,\ell+1)} = \left(\boldsymbol{A}_q^{(i,j)} + \boldsymbol{P}_q^{(i,j)}\right)^{-1} \left(\boldsymbol{R}^{(i,j)} \bar{\boldsymbol{\alpha}}^{(i,j,\ell)} + \boldsymbol{M}_q^{(i,j)} \boldsymbol{\alpha}_q^{(i^*,j^*,\ell+1)} + \boldsymbol{B}_q^{(i,j)} + \boldsymbol{S}_q^{(i,j)}\right), \tag{3.33a}$$

$$\bar{\boldsymbol{\alpha}}^{(i,j,\ell+1)} = \sum_{q=1}^{N} w_q \boldsymbol{\alpha}_q^{(i,j,\ell+1)}. \tag{3.33b}$$

Sweeping refers to process solving of (3.33) cell-by-cell: for each q, cells can be ordered such that $\boldsymbol{\alpha}_q^{(i^*,j^*,\ell+1)}$ is known, prior to solving for $\boldsymbol{\alpha}_q^{(i,j,\ell+1)}$. The details of this procedure are given in Algorithm 4.

3.4.4 Monte Carlo

In this section, we describe the Monte Carlo method used to compute the solution to (3.7) for the pure absorption problem when $\lambda_\mathrm{t} = \lambda_\mathrm{a}$ (i.e. no scattering):

$$\begin{cases} \partial_t u + \boldsymbol{\Omega} \cdot \nabla_{\boldsymbol{x}} u + \lambda_\mathrm{t} u = s, & \boldsymbol{x} \in X, \quad \boldsymbol{\Omega} \in \mathbb{S}^2, \quad t > 0, & (3.34a) \\ u(\boldsymbol{x}, \boldsymbol{\Omega}, 0) = v(\boldsymbol{x}, \boldsymbol{\Omega}) & \boldsymbol{x} \in X, \quad \boldsymbol{\Omega} \in \mathbb{S}^2, & (3.34b) \\ u(\boldsymbol{x}, \boldsymbol{\Omega}, t) = b(\boldsymbol{x}, \boldsymbol{\Omega}, t) & \boldsymbol{x} \in \partial X, \quad \boldsymbol{\Omega} \cdot \boldsymbol{n}(\boldsymbol{x}) < 0, \quad t > 0. & (3.34c) \end{cases}$$

The Monte Carlo method approximates the phase space distribution u using a finite set of pseudo-particles:

$$u(\boldsymbol{x}, \boldsymbol{\Omega}, t) \approx \sum_{\pi \in \Pi^t} w_\pi(t) \delta(\boldsymbol{x} - \boldsymbol{x}_\pi(t)) \delta(\boldsymbol{\Omega} - \boldsymbol{\Omega}_\pi), \tag{3.35}$$

where Π^t is a set of pseudo-particles π with position $x_\pi(t)$, weight $w_\pi(t)$, and direction of flight $\boldsymbol{\Omega}_\pi$. Π^t will be defined more carefully below. A benefit of the hybrid is that scattering processes, which can slow down the method significantly [13], do not need to be modeled in (3.34).

The Monte Carlo implementation of (3.34) can be derived via a Green's function formulation for (3.34). Let $G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\Omega}, t, t_0)$ solve

$$\partial_t G + \boldsymbol{\Omega} \cdot \nabla_x G + \lambda_t G = \delta(\boldsymbol{x} - \boldsymbol{y}) \delta(t - t_0), \tag{3.36}$$

with zero initial data and boundary conditions. Then

$$\begin{aligned} u(\boldsymbol{x}, \boldsymbol{\Omega}, t) = &\int_0^t \int_{\mathbb{R}^3} G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\Omega}, t, t_0) s(\boldsymbol{y}, \boldsymbol{\Omega}, t_0) d\boldsymbol{y} dt_0 && \text{(3.37a)} \\ &+ \int_0^t \int_{\mathbb{R}^3} G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\Omega}, t, t_0) s_v(\boldsymbol{y}, \boldsymbol{\Omega}, t_0) d\boldsymbol{y} dt_0 && \text{(3.37b)} \\ &+ \int_0^t \int_{\mathbb{R}^3} G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\Omega}, t, t_0) s_b(\boldsymbol{y}, \boldsymbol{\Omega}, t_0) d\boldsymbol{y} dt_0 && \text{(3.37c)} \end{aligned}$$

solves (3.34), where s, s_v, and s_b are identically zero outside of the closure of X. The terms s_v and s_b are provisional source terms designed such that (3.37b) solves (3.34) when $s = 0$ and $b = 0$, while (3.37c) solves (3.34) when $s = 0$ and $v = 0$. The former is solved by setting $s_v(\boldsymbol{x}, \boldsymbol{\Omega}, t) = v(\boldsymbol{x}, \boldsymbol{\Omega}) \delta(t)$. However, determining s_b can be slightly more involved, and an example that is used for numerical experiments in Section 3.5.3 is provided in the Appendix. Once s, s_v, and s_b are known, all three terms can be treated identically. For simplicity, we restrict our attention to (3.37a) below.

For any $t > t_0$ and any fixed $\boldsymbol{\Omega}$, let $\boldsymbol{X}(t) = \boldsymbol{x}_0 + (t - t_0)\boldsymbol{\Omega}$. Then $g(t) = G(\boldsymbol{X}(t), \boldsymbol{y}, \boldsymbol{\Omega}, t, t_0)$ solves

$$\frac{dg(t)}{dt} = -\lambda(\boldsymbol{X}(t))g(t) + \delta(\boldsymbol{X}(t) - \boldsymbol{y})\delta(t - t_0) \tag{3.38}$$

or, equivalently,

$$g(t) = e^{-\int_{t_0}^{t} \lambda_{\mathrm{t}}(\boldsymbol{X}(\xi))d\xi}\delta(\boldsymbol{x}_0 - \boldsymbol{y}). \tag{3.39}$$

Setting $\boldsymbol{x} = \boldsymbol{X}(t)$ in (3.39) gives

$$G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\Omega}, t, t_0) = e^{-\int_{t_0}^{t} \lambda_{\mathrm{t}}(\boldsymbol{x} - (t-\xi)\boldsymbol{\Omega})d\xi}\delta(\boldsymbol{x} - (t - t_0)\boldsymbol{\Omega} - \boldsymbol{y}). \tag{3.40}$$

Plugging (3.40) back into (3.37a) yields, after some manipulation,

$$\begin{aligned} u(\boldsymbol{x}, \boldsymbol{\Omega}, t) &= \int_0^t \int_{\mathbb{R}^3} e^{-\int_{t_0}^{t} \lambda_{\mathrm{t}}(\boldsymbol{x} - (t-\xi)\boldsymbol{\Omega})d\xi}\delta(\boldsymbol{x} - (t - t_0)\boldsymbol{\Omega} - \boldsymbol{y})s(\boldsymbol{y}, \boldsymbol{\Omega}, t_0)d\boldsymbol{y}dt_0 \\ &= \int_0^t e^{-\int_{t_0}^{t} \lambda_{\mathrm{t}}(\boldsymbol{x} - (t-\xi)\boldsymbol{\Omega})d\xi} s(\boldsymbol{x} - (t - t_0)\boldsymbol{\Omega}, \boldsymbol{\Omega}, t_0)dt_0 \\ &= \int_0^t e^{-\int_{t-\tau}^{t} \lambda_{\mathrm{t}}(\boldsymbol{x} - (t-\xi)\boldsymbol{\Omega})d\xi} s(\boldsymbol{x} - \tau\boldsymbol{\Omega}, \boldsymbol{\Omega}, t - \tau)d\tau \\ &= \int_0^t e^{-\int_{0}^{\tau} \lambda_{\mathrm{t}}(\boldsymbol{x} - \xi\boldsymbol{\Omega})d\xi} s(\boldsymbol{x} - \tau\boldsymbol{\Omega}, \boldsymbol{\Omega}, t - \tau)d\tau. \end{aligned} \tag{3.41}$$

This representation of $u(\boldsymbol{x}, \boldsymbol{\Omega}, t)$ can be interpreted as the density of particles that have reached the location $\boldsymbol{x}$ at time t while moving in the direction $\boldsymbol{\Omega}$. These particles are emitted by the source s at time $t - \tau$ and location $\boldsymbol{x} - t\boldsymbol{\Omega}$, and they carry a weight that decays exponentially due to absorption.

The Monte Carlo approach can be understood as an approximation of u based on sampling of pseudo-particles from the source s in (3.41). Let

$$\tilde{s}(\boldsymbol{x}, \boldsymbol{\Omega}, t) = \sum_{\pi \in \Pi} w_\pi \delta(\boldsymbol{x} - \boldsymbol{x}_\pi)\delta(\boldsymbol{\Omega} - \boldsymbol{\Omega}_\pi)\delta(t - t_\pi) \tag{3.42}$$

where $\pi \in \Pi$ are pseudo-particles with weight $w_\pi > 0$, position $\boldsymbol{x}_\pi \in X$, and direction of flight $\boldsymbol{\Omega}_\pi \in \mathbb{S}^2$ at $t_\pi > 0$, such that $\tilde{s}(\boldsymbol{x}, \boldsymbol{\Omega}, t)$ approximates $s(\boldsymbol{x}, \boldsymbol{\Omega}, t)$ for all $t > 0$. Then for any $C \subset X$, any $B \subset \mathbb{S}^2$, and any time interval (t_n, t_{n+1}), the representation of u in (3.41), along with the approximation $\tilde{s}$ gives

$$
\begin{aligned}
&\int_{t_n}^{t_{n+1}} \int_C \int_B u(\boldsymbol{x}, \boldsymbol{\Omega}, t) d\boldsymbol{\Omega} d\boldsymbol{x} dt \\
&\approx \int_{t_n}^{t_{n+1}} \int_C \int_B \int_0^t e^{-\int_0^\tau \lambda_{\mathrm{t}}(\boldsymbol{x}-\xi\boldsymbol{\Omega})d\xi} \tilde{s}(\boldsymbol{x} - \tau\boldsymbol{\Omega}, \boldsymbol{\Omega}, t-\tau) d\tau d\boldsymbol{\Omega} d\boldsymbol{x} dt \\
&= \int_{t_n}^{t_{n+1}} \int_C \int_B \int_0^t e^{-\int_0^\tau \lambda_{\mathrm{t}}(\boldsymbol{x}-\xi\boldsymbol{\Omega})d\xi} \sum_{\pi\in\Pi} w_\pi \delta(\boldsymbol{x} - \tau\boldsymbol{\Omega} - \boldsymbol{x}_\pi)\delta(\boldsymbol{\Omega} - \boldsymbol{\Omega}_\pi)\delta(t - \tau - t_\pi) d\tau d\boldsymbol{\Omega} d\boldsymbol{x} dt \\
&= \sum_{\pi\in\Pi} \int_{t_n}^{t_{n+1}} w_\pi e^{-\int_0^{t-t_\pi} \lambda_{\mathrm{t}}(\boldsymbol{x}_\pi+\xi\boldsymbol{\Omega}_\pi)d\xi} \mathbb{1}_C(\boldsymbol{x}_\pi + (t-t_\pi)\boldsymbol{\Omega}_\pi) \mathbb{1}_{[0,t]}(t_\pi) \mathbb{1}_B(\Omega_\pi) dt \\
&= \sum_{\pi\in\Pi} \int_{t_n}^{t_{n+1}} w_\pi(t) \mathbb{1}_C(\boldsymbol{x}_\pi + (t-t_\pi)\boldsymbol{\Omega}_\pi) \mathbb{1}_{[0,t]}(t_\pi) \mathbb{1}_B(\Omega_\pi) dt
\end{aligned}
\tag{3.43}
$$

where $w_\pi(t) = w_\pi e^{-\int_0^{t-t_\pi} \lambda_{\mathrm{t}}(\boldsymbol{x}_\pi+\xi\boldsymbol{\Omega}_\pi)d\xi}$ and $w_\pi(t_\pi) = w_\pi$.

Note that with the identification $\boldsymbol{x}_\pi(t) = \boldsymbol{x}_\pi + (t - t_\pi)\boldsymbol{\Omega}_\pi$ we can also identify Π^t in (3.35) as $\Pi^t = \{\pi \in \Pi : t_\pi < t\}$.

We will denote (3.43) as $\mathcal{T}_{\mathrm{MC}}(t_{n+1}, t_n, 0, \lambda_{\mathrm{s}}, \lambda_{\mathrm{s}}, s, 0; N_p)$ as the Monte Carlo solution for the case of the zero initial data and zero boundary data. A general Monte-Carlo solution can be obtained as

$$
\begin{aligned}
\mathcal{T}_{\mathrm{MC}}(t_{n+1}, t_n, s, v, b, \lambda_{\mathrm{s}}, \lambda_{\mathrm{s}}; N_p) =& \mathcal{T}_{\mathrm{MC}}(t_{n+1}, t_n, s + s_b, v, 0, \lambda_{\mathrm{s}}, \lambda_{\mathrm{s}}; N_p) \\
=& \mathcal{T}_{\mathrm{MC}}(t_{n+1}, t_n, s + s_v + s_b, 0, 0, \lambda_{\mathrm{s}}, \lambda_{\mathrm{s}}; N_p)
\end{aligned}
$$

Numerically it is useful to realize that

$$
\Pi^{n+1} := \Pi^{t_{n+1}} = \left\{\pi(t_n + \Delta t) : \pi \in \Pi^{t_n}\right\} \cup \{\pi \in \Pi : t_\pi \in (t_n, t_{n+1})\} \tag{3.44}
$$

where in an abuse of notation $\pi(t_n + \Delta t)$ is a new particle with position $x_\pi(t + \Delta t)$, weight $w_\pi(t + \Delta t)$ and direction of flight $\boldsymbol{\Omega}_\pi$ for some $\pi \in \Pi^{t_n} = \Pi^n$. Thus particles Π^{n+1} can be obtained by updating the weight and position of particles in Π^n and sampling new particles

with birth times in (t_n, t_{n+1}). From (3.43) we also obtain

$$\langle u \rangle_{\mathrm{MC}}(\boldsymbol{x}, t) = \sum_{\pi \in \Pi} \int_{t_n}^{t_{n+1}} w_\pi(t) \mathbb{1}_C(\boldsymbol{x}_\pi + (t - t_\pi)\boldsymbol{\Omega}_\pi) \mathbb{1}_{[0,t]}(t_\pi) dt \tag{3.45}$$

With this formulation, particle weights $w_\pi(t)$ decrease exponentially at a rate proportional to the absorption, given by λ_{t}. This approach is known as the implicit capture method. It avoids the need to sample absorption times explicitly and, at the same time, reduces statistical noise and simplifies the implementation [14, Page 168][26, Chapter 22].

The Monte Carlo simulation of (3.34) from t_n to t_{n+1} is based on (3.43) and proceeds according to the following steps:

1. Let $\mathcal{P}$ be a partition of X into disjoint cells C. For each $C \in \mathcal{P}$, calculate the total weight W^C of new particles generated in C by the source during the interval (t_n, t_{n+1}):

$$W^C = \frac{1}{\Delta t} \frac{1}{4\pi} \int_{\mathbb{S}^2} \int_{t_n}^{t_{n+1}} \int_C s(\boldsymbol{x}, \Omega, t)\, d\Omega d\boldsymbol{x} dt, \tag{3.46}$$

   where $\Delta t = t^{n+1} - t^n$, and let $W = \sum_{C \in \mathcal{P}} W^C$. Let N_p be the input for the total number of new particles to sample during the interval (t_n, t_{n+1}). Then for each $C \in \mathcal{P}$, sample

$$N_p^C = \text{floor}\left[\frac{W^C}{W}\right] \tag{3.47}$$

   particles and assign them each a weight $w = W/N_p^C$. The total number of particles in the system at this time is $\tilde{N}_p^{n+1} = \tilde{N}_p^{n+1} + \sum_{C \in \mathcal{P}} N_p^C$. Each new particle p is assigned a position $\boldsymbol{x}_\pi$ sampled uniformly from C and a birth time $t^{n+1} - \tau_\pi$, where τ_π is sampled uniformly from $(0, \Delta t)$. Each particle p is also assigned an angle $\boldsymbol{\Omega}_\pi$. For isotropic sources, $\boldsymbol{\Omega}_\pi$ is sampled uniformly from $\mathbb{S}^2$. For non-isotropic sources, (such as the boundary source for the holhraum problem in Section 3.5.3), the sampling distribution must be consistent with the angular dependence. The particles, including their space, angle, and time coordinates, are added to the current particle list.

2. Move each particle π in the current particle list from $\boldsymbol{x}_\pi$ to $x_\pi + \tau_\pi \boldsymbol{\Omega}_\pi$ and update its weight to $w_\pi \leftarrow w_\pi(t_{n+1})$. The number of particles in the system will have to be

adjusted accordingly. Reset its remaining time to $\tau_\pi \leftarrow \Delta t$. For each cell $C \in \mathcal{P}$, update the sum in (3.43) and (3.45) according the time spent in C during the interval (t_n, t_{n+1}).

3. To reduce computational effort and memory, at the end of each time step any particle p with weight $w_\pi < w_{\text{kill}}$ will be dropped with a probability of $p_{\text{kill}} > 0$. Here $w_{\text{kill}} > 0$ is a user-defined parameter, called the 'killing weight', and $p_{\text{kill}} = (1 - w_\pi / w_{\text{kill}})$. To preserve the total mass in the system, any particle p with weight $w < w_{\text{kill}}$ that survives this 'Russian roulette' [26, Chapter 22] will have its weight readjusted to $w_\pi/(1 - p_{\text{kill}})$.

3.4.5 Pseudocode

The algorithms that we use for our numerical results are detailed in Algorithms 4-7. The S_N method is given in Algorithm 4. The hybrid method is given in Algorithm 7. It requires Algorithm 4 for the collided component and Algorithm 5 for the Monte Carlo update. A listing of the notation used these algorithms is provided in Table 3.1.

3.5 Numerical Results

In this section, we compare simulation results from the Monte Carlo-S_N hybrid method to those from a standard, monolithic S_N method. The goal is to demonstrate that the hybrid method provides a more efficient approach. The S_N computations for the monolithic method and for the collided component of the hybrid rely on product quadrature sets on the sphere [40, 4].

We consider three well known test problems: the line source problem [17], the lattice problem [8], and the linearized hohlraum problem [9, 8]. The specifications for each problem are provided in the following subsections. They are all formulated in a geometry for which $\partial_z \Psi = 0$. This means that they can be reduced to two dimensions in physical space and, by an abuse of notation, we write $\Psi(\boldsymbol{x}, \boldsymbol{\Omega}, t) = \Psi(x, y, \boldsymbol{\Omega}, t)$. A further consequence of the geometry is that product quadrature on the sphere can be reduced to just the upper hemisphere, in which case $N_{\boldsymbol{\Omega}} = N^2$. The time step for each problem is tied to the grid resolution via

Table 3.1: Pseudocode parameters (In text mentions: p.67)

User defined parameters	
$N_{\boldsymbol{x}}$	number of Cartesian cells along each dimension
N	order of discrete ordinates
N_p	number of new particles (up to rounding) generated from source
Δt	time step
δ	tolerance of iteration
ω_q	Gauss-Legendre weights
Material parameters	
$\lambda_{\mathrm{t}}, \lambda_{\mathrm{a}}, \lambda_{\mathrm{s}}$	total, absorption and scattering crosssection
Additional notation	
$\boldsymbol{A}_q^{(i,j)}, \boldsymbol{P}_q^{(i,j)}, \boldsymbol{R}^{(i,j)}, \boldsymbol{M}_q^{(i,j)}, \boldsymbol{B}_q^{(i,j)}, \boldsymbol{S}_q^{(i,j)}$	matrices defined in Section 3.4.3
$\mathcal{U}(X)$	uniform distribution on set X.

Algorithm 4 S_N-algorithm: Propagate solution from t_k to t_{k+1}

Input: $u^n, \mathbf{s}_q$ ▷ coefficients of solution u_q^k from previous step and source

Require: $\hat{\lambda}_t, \lambda_a, \lambda_s,$ ▷ Material properties

Require: $\Delta t, \{C_{i,j}\}_{i,j=1}^{N_{\boldsymbol{x}}}, \{\boldsymbol{\Omega}_q, w_q\}_{q=1}^{N_{\boldsymbol{\Omega}}}$ ▷ Discretization parameters

Require: δ ▷ Convergence tolerance

$\boldsymbol{\alpha}_q^{(i,j)}(t_n)$ such that $u_q^n(\boldsymbol{x}) = \sum_{|\boldsymbol{k}|_\infty \leq 1} \alpha_{q,\boldsymbol{k}}^{(i,j)} \phi_{\boldsymbol{k}}^{(i,j)}(\boldsymbol{x})$ for $\boldsymbol{x} \in C_{i,j}$

$\boldsymbol{s}_q \leftarrow \boldsymbol{s}_q + \frac{1}{\Delta t}\boldsymbol{\alpha}_q(t_n)$ ▷ Initialize source

$\boldsymbol{\alpha}_q \leftarrow \boldsymbol{\alpha}_q(t_n)$ ▷ Initialize coefficients

$err = \delta + 1$

while $err > \delta$ **do**

  $\boldsymbol{\beta}_q \leftarrow \boldsymbol{\alpha}_q$ ▷ Store old coefficients

  $\bar{\boldsymbol{\alpha}} \leftarrow \sum_{q=1}^{N_{\boldsymbol{\Omega}}} w_q \boldsymbol{\alpha}_q$

  for $q \in \{1, ..., N_{\boldsymbol{\Omega}}\}$ **do**

    for $(i,j) \in \{1, ..., N_{\boldsymbol{x}}\}^2$ **do** ▷ Sweep through cells in direction $\boldsymbol{\Omega}_q$

      $\boldsymbol{\alpha}_q^{(i,j)} \leftarrow \left(\boldsymbol{A}_q^{(i,j)} + \boldsymbol{P}_q^{(i,j)}\right)^{-1} \left(\boldsymbol{R}^{(i,j)}\bar{\beta}^{(i,j)} + \boldsymbol{M}_q^{(i,j)}\boldsymbol{\alpha}_q^{(i^*,j^*)} + \boldsymbol{B}_q^{(i,j)} + \boldsymbol{S}_q^{(i,j)}\right)$ ▷ Update coefficients

    end for

  end for

  $err = \max_q ||\boldsymbol{\alpha}_q - \boldsymbol{\beta}_q||_\infty$ ▷ Discrepancy between iterations

end while

return $u_q^{n+1}(\boldsymbol{x}) = \sum_{|\boldsymbol{k}|_\infty \leq 1} \alpha_{q,\boldsymbol{k}}^{(i,j)} \phi_{\boldsymbol{k}}^{(i,j)}(\boldsymbol{x}), \langle u_q^{n+1} \rangle_{\mathrm{SN}}(\boldsymbol{x}) = \sum_{|\boldsymbol{k}|_\infty \leq 1} \bar{\alpha}_{\boldsymbol{k}}^{(i,j)} \phi_{\boldsymbol{k}}^{(i,j)}(\boldsymbol{x})$ for $\boldsymbol{x} \in C_{i,j}$

Algorithm 5 MC-algorithm: Propagate solution form t_k to t_{k+1}

Input: Π^{n+1} with $u^n(\boldsymbol{x}, \boldsymbol{\Omega}) = \sum_{\pi \in \Pi^t} w_\pi \delta(\boldsymbol{x} - \boldsymbol{x}_\pi)\delta(\boldsymbol{\Omega} - \boldsymbol{\Omega}_\pi)$

$s(\boldsymbol{x}, \boldsymbol{\Omega}, t)$ ▷ previous particle distribution and source

Require: λ_{t}

Require: $\Delta t, \{C_{i,j}\}_{i,j}^{N_{\boldsymbol{x}}}, N_p$ ▷ Discretization parameters

Require: w_{kill} ▷ killing weight

for $(i,j) \in \{1, \dots, N_x\}^2$ **do**

  $W^{C_{i,j}} = \frac{1}{|C_{i,j}|\Delta t} \int_{t_n}^{t_{n+1}} \int_{C_{i,j}} \int_{\mathbb{S}^2} s(\boldsymbol{x}, \boldsymbol{\Omega}, t) d\boldsymbol{\Omega} d\boldsymbol{x} dt$

end for

$W \leftarrow \sum_{i,j} W^{C_{i,j}}$ ▷ Calculate total source

for $(i,j) \in \{1, \dots, N_x\}^2$ **do**

  $N_p^{C_{i,j}} \leftarrow \left\lfloor \frac{W^{C_{i,j}} N_p}{W} \right\rfloor$ ▷ Number of particles on each cell

end for

$N_p \leftarrow \sum_{i,j} N_p^{C_{i,j}}$ ▷ number of new particles

$w \leftarrow \frac{W}{N_p}$

for (i,j) with $N_p^{C_{i,j}} > 0$ **do** ▷ sample new particles from source

  for $k \in \{1, ..., N_p^{C_{i,j}}\}$ **do**

    Generate new particle π with

    $\boldsymbol{x}_\pi \sim \mathcal{U}(C_{i,j})$ ▷ Draw particle's position

    $(\Omega_x, \Omega_y, \Omega_z) \sim \frac{1}{W^{C_{i,j}}|C_{i,j}|\Delta t} \int_{t_n}^{t_{n+1}} \int_{C_{i,j}} s(\boldsymbol{x}, \boldsymbol{\Omega}, t) d\boldsymbol{x} dt$ ▷ Draw particle's direction of flight

    $\boldsymbol{\Omega}_\pi \leftarrow (\Omega_x, \Omega_y)$

    $w_\pi \leftarrow w$ ▷ Assign particle weight

    $\Pi^* \leftarrow \Pi^* \cup \{\pi\}$ ▷ Add new particle to existing

  end for

end for

for $\pi \in \Pi^t \cup \Pi^*$ **do** ▷ Move particles

  if $\pi \in \Pi^t$ **then**

    $\tau_\pi \leftarrow \Delta t$ ▷ remaining time for particles from prev. step

  else

    randomly draw $\tau_\pi \sim \mathcal{U}([0, \Delta t])$ ▷ remaining time for particles

  end if

  $\boldsymbol{x}_\pi \leftarrow \boldsymbol{x}_\pi + \tau_\pi \boldsymbol{\Omega}_\pi$ ▷ Update particle's position

  for (i,j) with $C_{i,j} \cap \{\boldsymbol{x}\pi + t\boldsymbol{\Omega}_\pi : t \in [0, \tau_\pi]\} \neq \emptyset$ **do** ▷ All cells intersected by particle's trajectory

    $\Phi_{i,j} \leftarrow \Phi_{i,j} + w_\pi \int_0^{\tau_\pi} \exp\left(-\int_0^t \lambda_{\mathrm{a}}(\boldsymbol{x}_p + t'\boldsymbol{\Omega}_p) dt'\right) \mathbb{1}_{C_{i,j}}(\boldsymbol{x}_\pi + t\boldsymbol{\Omega}_\pi) dt$ ▷ Update Φ

  end for

  $w_\pi \leftarrow w_\pi \exp\left(-\int_0^{\tau_\pi} \lambda_{\mathrm{a}}(\boldsymbol{x}_\pi + t\boldsymbol{\Omega}_p) dt\right)$ ▷ Update particle weight

  if $\boldsymbol{x}_\pi \in X$ **then**

    $\Pi^{n+1} \leftarrow \Pi^{n+1} \cup \{\pi\}$ ▷ Remove particles that left domain

  end if

end for

Continuation of Algorithm 5 (Russian Roulette)

for $\pi \in \Pi^{n+1}$ with $w_\pi < w_{\text{kill}}$ **do** ▷ Russian roulette
  $r \sim \mathcal{U}([0,1])$
  if $r > \frac{w_\pi}{w_{\text{kill}}}$ **then** ▷ Determine survival of particle
    $\Pi^{n+1} \leftarrow \Pi^{n+1} \setminus \{\pi\}$
  else
    $w_\pi \leftarrow w_{\text{kill}}$ ▷ Update surviving particle's weight to approx. preserve total mass
  end if
end for
return Π^{n+1} with $u^{n+1}(\boldsymbol{x}, \boldsymbol{\Omega}) = \sum_{\pi \in \Pi^{n+1}} w_\pi \delta(\boldsymbol{x} - \boldsymbol{x}_\pi)\delta(\boldsymbol{\Omega} - \boldsymbol{\Omega}_\pi)$ and $\langle u^{n+1} \rangle_{\text{MC}}(\boldsymbol{x}) = \Phi(\boldsymbol{x})$

Algorithm 7 $\text{H}_{\text{MC-S}_\text{N}}$-algorithm: Propagate solution from t_k to t_{k+1}

Input: Π^n with $u^n(\boldsymbol{x}, \boldsymbol{\Omega}) = \sum_p w_p \delta(\boldsymbol{x} - \boldsymbol{x}_p)\delta(\boldsymbol{\Omega} - \boldsymbol{\Omega}_p)$
$s(\boldsymbol{x}, \boldsymbol{\Omega}, t)$ ▷ previous particle distribution and source
Require: $\hat{\lambda}_\text{t}, \lambda_\text{t}, \lambda_\text{a}, \lambda_\text{s},$ ▷ Material properties
Require: $\Delta t, \{C_{i,j}\}_{i,j=1}^{N_{\boldsymbol{x}}}, \{\boldsymbol{\Omega}_q, w_q\}_{q=1}^{N_{\boldsymbol{\Omega}}}, N_p$ ▷ Discretization parameters
Require: δ, w_{kill} ▷ Convergence tolerance and killing weight
$\Pi_u^{n+1}, \langle u_u^{n+1} \rangle_{\text{MC}} \leftarrow \text{MC}(\Pi^n, s)$ ▷ Monte Carlo for uncollided
for $q \in \{1, ..., N_{\boldsymbol{\Omega}}\}$ **do**
  $\boldsymbol{s}_q \leftarrow \lambda_\text{s} \Phi_u$ ▷ turn $\langle u_u^{n+1} \rangle_{\text{MC}}$ into source for S_n
end for
$u_c^{n+1}, \langle u_c^{n+1} \rangle_{\text{SN}} \leftarrow \text{S}_\text{n}(\boldsymbol{0}, \boldsymbol{s}_q)$ ▷ S_n for collided
$\boldsymbol{s} \leftarrow \lambda_\text{s}(\langle u_c^{n+1} \rangle_{\text{SN}} + \langle u_u^{n+1} \rangle_{\text{MC}})$ ▷ sources for Relabeling
$\Pi_R, \langle u_R^{n+1} \rangle_{\text{MC}} \leftarrow \text{MC}(0, \boldsymbol{s})$ ▷ Monte Carlo as relabeling
$\Pi^{n+1} \leftarrow \Pi_u^{n+1} \cup \Pi_R$
$\langle u^{n+1} \rangle_{\text{MC}} \leftarrow \langle u_u^{n+1} \rangle_{\text{MC}} + \langle u_R^{n+1} \rangle_{\text{MC}}$
return $\Pi^{n+1}, \langle u^{n+1} \rangle_{\text{MC}}$

the ratio CFL $= \Delta t/\Delta x$. For all calculations shown below, the iteration tolerance δ (see Algorithm 1) is set to 10^{-4}.[1]

For each problem, we assess the accuracy of the numerical solution and the efficiency with which it is obtained. To quantify the accuracy we compare our results to a reference solution. For the line source problem the reference is the semi-analytic solution from [18]; see also [17]. For the other two test problems, the reference is a high-resolution hybrid solution based solely on S_N discretization with a triangular-based quadrature referred to as T_N [37] for both collided and uncollided component, combined with a DG discretization in space and integral deferred correction in time [11].

Accuracy for the MC-S_N hybrid and the monolithic S_N method is measured in terms of the relative L^2-difference in the scalar flux $\Phi = \langle \Psi \rangle$ at a given final time t_{final}. Given the numerical solution Φ_{num} and the reference Φ_{ref}:

$$\Delta = \frac{||\Phi - \Phi_{\text{ref}}||_{L_2}}{||\Phi_{\text{ref}}||_{L_2}}, \tag{3.48}$$

where L^2-norm is approximated by $h^2 \sum_{C_{i,j}} \Phi_{ij}^2$ and $\Phi_{i,j}$ is the average on the cell $C_{i,j}$. Because our implementation of the hybrid method and the S_N method are not run-time optimized, we use a complexity measure which counts the number of times a particle is moved or a DG unknown is updated in the course of a sweep. Let N^{c} be the level of the quadruature for the collided component of the hybrid. Then the complexity of the monolithic method is

$$\mathbb{C}_{S_N} = \underset{\substack{\uparrow \\ \#\text{ of} \\ \text{Legendre} \\ \text{coefficients}}}{4} \times \underset{\substack{\uparrow \\ \#\text{ of} \\ \text{ordi-} \\ \text{nates}}}{N_{\mathbf{\Omega}}} \times \underset{\substack{\uparrow \\ \#\text{ of} \\ \text{cells}}}{N_{\boldsymbol{x}}^2} \times \underset{\substack{\uparrow \\ \#\text{ of} \\ \text{source} \\ \text{iterations}}}{N_i} \times \underset{\substack{\uparrow \\ \#\text{ of} \\ \text{time} \\ \text{steps}}}{\frac{T}{\Delta t}} \tag{3.49a}$$

[1] While not shown here, we also tested several runs using $\delta = 10^{-8}$, which leads to negligible improvements in accuracy when compared to the results shown below.

while the complexity of the hybrid is

$$\mathbb{C}_{\text{hybrid}} = \underset{\substack{\uparrow \\ \text{avg. \# of particles moved for uncollided}}}{(N_u} + \underset{\substack{\uparrow \\ \text{avg. \# particles moved in relabelling}}}{N_R)} \times \underset{\substack{\uparrow \\ \text{\# of time steps}}}{\frac{T}{\Delta t}} + \underset{\substack{\uparrow \\ S_N \text{ complexity of collided equation}}}{\mathbb{C}_{S_N\text{c}}} \tag{3.49b}$$

For convenience, we set $N_{\text{MC}}^{\text{tot}} = (N_u + N_R)\frac{T}{\Delta t}$ so that $\mathbb{C}_{\text{hybrid}} = N_{\text{MC}}^{\text{tot}} + \mathbb{C}_{S_N\text{c}}$. N_u is the sum of particles added to the system N_p and the average number of particles still in flight from the previous time step N_{prev}, i.e. $N_u = N_p + N_{prev}$, while N_R is the number of particles added in the relabeling, which we set to be $N_R = N_p$.

3.5.1 The Line Source problem

In the line source problem, an initial pulse of uniformly distributed particles is emitted from the line $\ell = \{(x, y, z) : x = y = 0\}$ into the surrounding domain $X = \mathbb{R}^3$, which contains a purely scattering material with $\sigma_{\text{s}} = \sigma_{\text{t}} = 1$. Because the geometry of the domain and initial condition are invariant in z, the spatial domain can be reduced to $\mathbb{R}^2$. In this two-dimensional setting, the initial condition can be represented by an isotropic delta function $\frac{1}{4\pi}\delta(x, y)$, but to reduce numerical artifacts, we use a mollified version of the initial condition:

$$\Psi_0(x, y, \boldsymbol{\Omega}, t) = \frac{1}{4\pi}\frac{1}{2\pi\varsigma}\exp\left(-\frac{x^2 + y^2}{2\varsigma}\right), \qquad \varsigma = 0.03. \tag{3.50}$$

Meanwhile, the computational domain is restricted to the square $[-1.5, 1.5]^2$ and equipped with zero inflow boundary conditions.

We perform monolithic S_N and MC-S_N hybrid simulations at various spatial and angular resolutions. The spatial domain is subdivided into equal $N_{\boldsymbol{x}} \times N_{\boldsymbol{x}}$ square cells with $N_{\boldsymbol{x}} \in \{51, 101, 201\}$. For the S_N-runs we let $N \in \{4, 8, 16, 32\}$, resulting in $N_{\boldsymbol{\Omega}} = N^2$ ordinates on the northern hemisphere of $\mathbb{S}^2$. The collided part of the hybrid algorithm employs an S_N

method with $N \in \{4, 8\}$. In the hybrid method the number of particles was also changed between runs.

At time $t = 0$ an initial pulse of N_p particles distributed according to (3.50) is added to the system. Since the source term is zero, new particles are only added due to relabeling. The number of particles newly inserted into the system is roughly N_p per time step where $N_p = 10^k$ and $k \in \{2, 3, 4, 5, 6\}$.

Due to rounding and particles being dropped via Russian roulette, the exact number of particles inserted into the system varies slightly. The killing weight is fixed at $w_{\text{kill}} = 10^{-15}$. The CFL is also fixed at 0.5 across all runs. The reference solution is the semi-analytic solution from [18]; see also [17].

Figure 3.1 depicts several approximations of the scalar flux Φ at $t_{\text{final}} = 1$, calculated using the S_N method and the hybrid method. The solutions calculated using the S_N method clearly show ray-effects that only get resolved after a significant increase in the angular resolution. No such effects are seen in the hybrid solutions. The hybrid solutions do contain some noise, as the particle count is relatively low, but they preserve the symmetry of the problem up to a reasonable error. Unlike the S_N-method, the hybrid is able to capture the wave front of unscattered particles travelling away from the center with speed 1.

Figure 3.2 shows L_2-errors of the numerical solutions in log scale; the same trends are apparent. While the hybrid solutions mainly suffer from noise due to the stochastic nature of the Monte Carlo method, the S_N method has strong ray-effects and struggles to capture the analytic solution at the wave front.

A more systematic analysis of the numerical results is presented in Figure 3.3. This plot shows the relative L_2-error Δ of various runs versus their respective computational complexity $\mathbb{C}$. For the hybrid method, increasing the angular resolution N in the collided component yields a marginal improvement at best. However, changes in the particle number N_p for the uncollided component have a significant impact. For the S_N method, on the other hand, increasing the angular resolution yields a significant improvement in the accuracy. For smaller values of N, increasing the spatial resolution may actually increase the error. This is especially apparent in Figure 3.3 for the S_4 and S_8 results. For a fixed angular resolution, additional spatial accuracy will begin to resolve the ray effect anomalies in the solution.

Conversely, S_N results with lower spatial resolution benefit from error cancellation due to the numerical diffusion smoothing ray effects. The hybrid may also have larger errors if the particles per cell is too low.

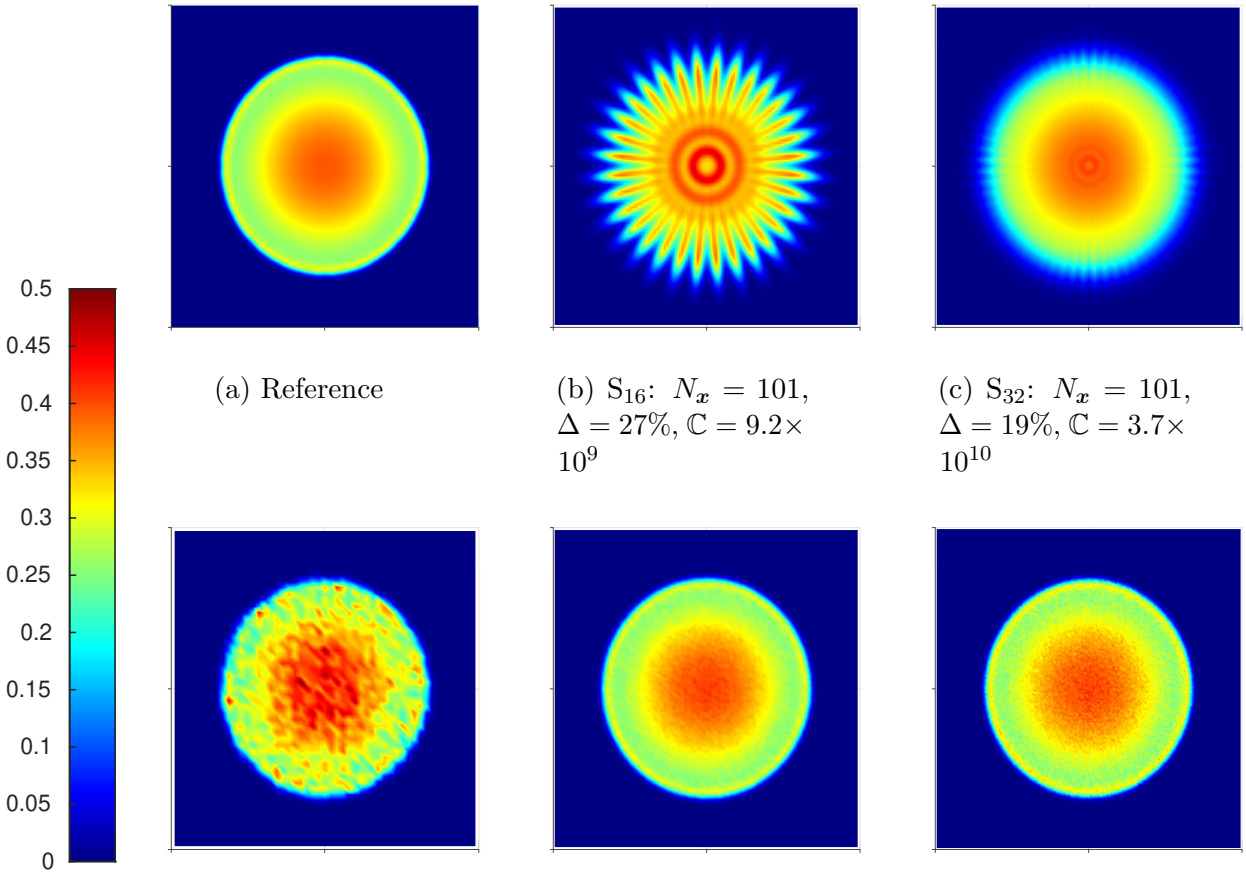
Overall the hybrid method outperforms the monolithic S_N method. For example, the hybrid error can match the most refined S_N calculation ($N = 32$) with a complexity that is roughly 2-3 orders of magnitude smaller; compare Figures 3.2 (c) and (d). In fact the hybrid method can obtain an error with half the size with a complexity that is an order of magnitude less. In general hybrid runs tend to be 3-4 times more accurate than their S_N counterparts of similar complexity.

3.5.2 The Lattice problem

In the lattice problem, a checkerboard of highly absorbing material is embedded in a scattering material with a central source. The layout of this problem along with its material parameters can be found in Figure 3.4. The computational domain is a 7×7 rectangle with zero inflow data at the boundaries. The center square (red) contains an isotropic particle source, while the blue squares are pure absorbers. The red and white squares are purely scattering with $\sigma_s = \sigma_t = 1$. The initial condition is identically zero everywhere in the domain.

We perform S_N and hybrid runs with varying spatial and angular resolution. The spatial domain is subdivided into equal $N_{\boldsymbol{x}} \times N_{\boldsymbol{x}}$ square cells with $N_{\boldsymbol{x}} \in \{56, 112, 224\}$. For the S_N-runs we use $N \in \{4, 8, 16, 32\}$, resulting in $N_{\boldsymbol{\Omega}} = N^2$ ordinates on the northern hemisphere of $\mathbb{S}^2$. The collided part of the hybrid algorithm employs an S_N method with $N \in \{4, 8\}$. In the hybrid method the number of particles is also changed between runs. The number of particles newly inserted into the system every time step is roughly $2 \times N_p$ where $N_p = 10^k$ for $k \in \{2, 3, 4, 5, 6\}$. [2] The killing weight is fixed at $w_{\text{kill}} = 10^{-15}$. All runs are performed to a final time of $t_{\text{final}} = 3.2$ and the CFL is kept fixed at 25.6. The reference solution is a S_{96}-S_{16} hybrid (meaning a S_{96} for the uncollided component and S_{16} for the collided component) using a T_N quadrature in angle, a third-order DG method in space on a 448 by 448 grid, and a defect correction time integrator [10].

[2] The factor of 2 is because N_p particles are used for the uncollided equations (3.13) and N_p particles are used for the relabelling (3.15).



(a) Reference

(b) S_{16}: $N_{\boldsymbol{x}} = 101$, $\Delta = 27\%$, $\mathbb{C} = 9.2\times 10^9$

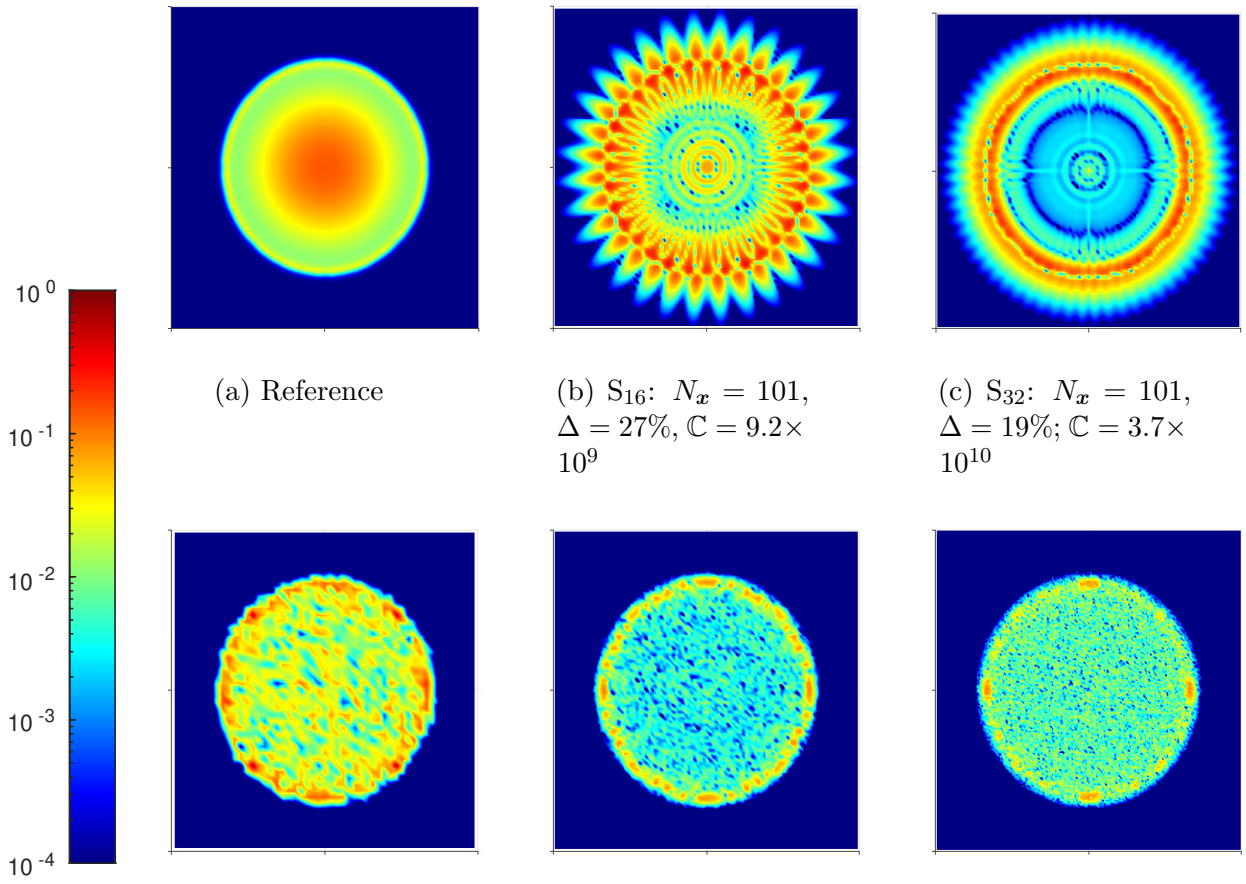
(c) S_{32}: $N_{\boldsymbol{x}} = 101$, $\Delta = 19\%$, $\mathbb{C} = 3.7\times 10^{10}$

(d) $H_{\text{MC-S}_4}$: $N_{\boldsymbol{x}} = 51$, $N_{\text{MC}}^{\text{tot}} = 5.9 \times 10^7$, $\Delta = 19\%$, $\mathbb{C} = 8.0 \times 10^7$

(e) $H_{\text{MC-S}_8}$: $N_{\boldsymbol{x}} = 101$, $N_{\text{MC}}^{\text{tot}} = 2.3 \times 10^9$, $\Delta = 8\%$; $\mathbb{C} = 2.5 \times 10^9$

(f) $H_{\text{MC-S}_4}$: $N_{\boldsymbol{x}} = 201$, $N_{\text{MC}}^{\text{tot}} = 9.0 \times 10^9$, $\Delta = 5\%$, $\mathbb{C} = 9.9 \times 10^9$

Figure 3.1: Numerical approximation of the scalar flux Φ for the line source problem at $t_{\text{final}} = 1$ with CFL 0.5. Each numerical solution is characterized by a relative L^2 difference Δ with respect to the reference, defined in (3.48), and a complexity $\mathbb{C}$, defined in (3.49a) for the monolithic S_N method and (3.49b) for the hybrid. (In text mentions: p.74)



(a) Reference

(b) S_{16}: $N_{\boldsymbol{x}} = 101$, $\Delta = 27\%$, $\mathbb{C} = 9.2\times 10^9$

(c) S_{32}: $N_{\boldsymbol{x}} = 101$, $\Delta = 19\%$; $\mathbb{C} = 3.7\times 10^{10}$

(d) $H_{\text{MC-S}_4}$: $N_{\boldsymbol{x}} = 51$, $N_{\text{MC}}^{\text{tot}} = 5.9 \times 10^7$, $\Delta = 19\%$, $\mathbb{C} = 8.0 \times 10^7$

(e) $H_{\text{MC-S}_8}$: $N_{\boldsymbol{x}} = 101$, $N_{\text{MC}}^{\text{tot}} = 2.3 \times 10^9$, $\Delta = 8\%$, $\mathbb{C} = 2.5 \times 10^9$

(f) $H_{\text{MC-S}_4}$: $N_{\boldsymbol{x}} = 201, N_{\text{MC}}^{\text{tot}} = 9.0 \times 10^9$ $\Delta = 5\%, \mathbb{C} = 9.9 \times 10^9$

Figure 3.2: Absolute difference between the analytical solution and various numerical solutions to the line source problem at $t = 1$ with CFL 0.5. Each numerical solution is characterized by a relative L^2 difference Δ with respect to the reference, defined in (3.48), and a complexity $\mathbb{C}$, defined in (3.49a) for the monolithic S_N method and (3.49b) for the hybrid. (In text mentions: pp.74,75)

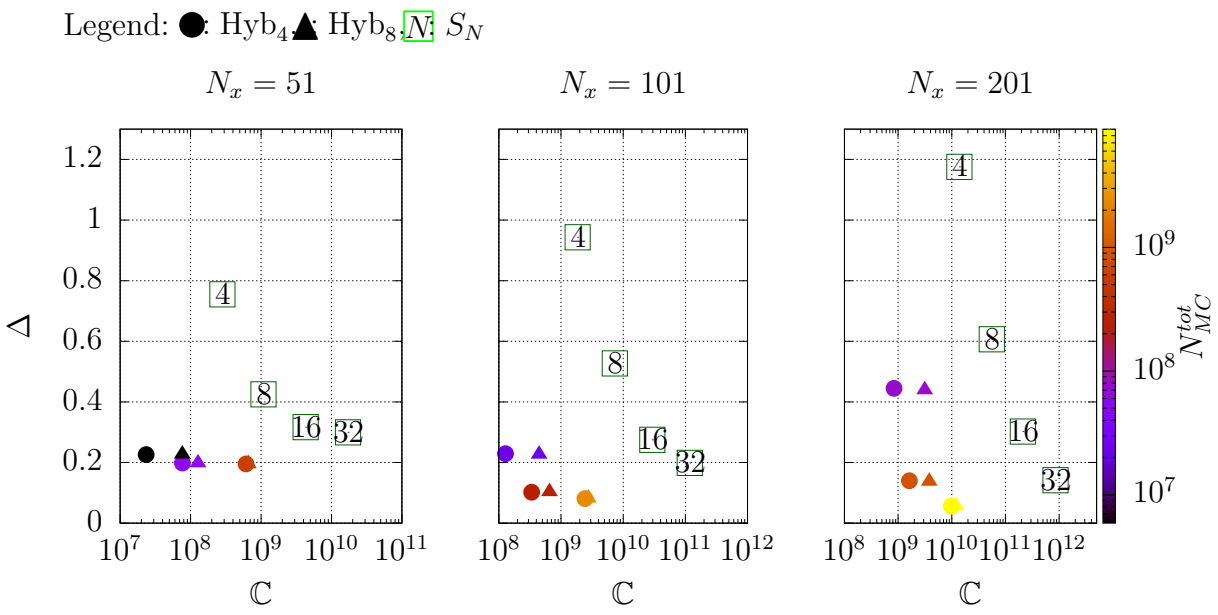


Figure 3.3: The relative L^2-difference Δ vs. complexity $\mathbb{C}$ for the scalar flux Φ in the line source problem, using the hybrid method with S_4(filled circle markers), the hybrid with S_8 (filled triangle markers) and the monolithic S_N method (empty, green markers). Points that are down and to the left are more efficient. All methods where run for three different spatial resolutions $N_{\boldsymbol{x}} = 51$(left), $N_{\boldsymbol{x}} = 101$(middle), $N_{\boldsymbol{x}} = 201$(right). Coloring of the hybrid data points corresponds to the total number of particles N_{MC}^{tot} according to the colorbar. The S_N method was run for $N = 4, 8, 16, 32$. Each DG-data point is assigned a numerical label according to its value of N. The formula for Δ is given in (3.48) which the complexity $\mathbb{C}$ is given by (3.49a) for the monolithic S_N method and by (3.49b) for the hybrid. (In text mentions: pp.74,74)

Selected results for the scalar flux Φ are depicted in Figure 3.5, and the relative L_2-error for these same solutions is depicted in Figure 3.6. While the hybrid solutions are not completely free of ray-effects, these effects are much more pronounced in the S_N runs. A more rigorous analysis of the performance of the two algorithm in dependence of their respective parameters can be seen in Figure 3.7. This plot shows the relative error Δ of various runs in dependence of their complexity $\mathbb{C}$. It is noteworthy that for this test problem the accuracy is mostly independent of the angular resolution, but depends significantly on the spatial resolution. Increasing the overall particle count in the hybrid method is most effective at higher spatial resolution; compare for example the vertical separation in colored triangles vs. colored circles vs. colored squares in Figure 3.7.

It turns out that increasing the number of particles in the hybrid algorithm does not necessarily increase the algorithm complexity. This is because with increased particle count the iterative solver for the collided components often needs fewer iterations. In cases where the complexity is dominated by these iterations, an increase in particles can even cause a decrease in computational complexity. Overall, Figure 3.7 shows that the hybrid algorithm produces results with comparable or slightly better accuracy than the standard S_N solver, while being close to an order of magnitude of lower complexity.

3.5.3 The linearized hohlraum problem

In the linearized hohlraum problem [9], nonlinear coupling between particles and the material medium is approximated in a linear way by adjusting the absorption and scattering cross-sections according to the expected material temperature profile of the nonlinear problem [8]. The geometry of the setup along with the material parameters can be found in Figure 3.8. The domain is $X = [0, 1.3] \times [0, 1.3]$, and the initial condition is identically zero everywhere. For boundary conditions, we assume a constant influx from the left side of the domain, i.e. $\Psi(x = 0, y, \boldsymbol{\Omega}, t) = 1$ for $\Omega_x > 0$. As discussed in the appendix, this boundary condition can be treated as a surface source, modeled by setting $\Omega_x = \sqrt{\xi}$, where $\xi \sim U([0, 1])$ is sampled uniformly on $[0, 1]$. The spatial distribution along the boundary is sampled uniformly.

We again perform S_N and hybrid runs with varying spatial and angular resolution. The spatial domain is subdivided into equal $N_{\boldsymbol{x}} \times N_{\boldsymbol{x}}$ square cells with $N_{\boldsymbol{x}} \in \{52, 104, 208\}$. For

the S_N-runs we use $N \in \{4, 8, 16, 32\}$, resulting in $N_{\mathbf{\Omega}} = N^2$ ordinates on the northern hemisphere of $\mathbb{S}^2$. The collided part of the hybrid algorithm employs an S_N method with $N \in \{4, 8\}$. In the hybrid method the number of particles is also changed between runs, but the killing weight remains fixed at $w_{\text{kill}} = 10^{-15}$. The number of particles newly inserted into the system every time step is 2×10^k for $k \in \{2, 3, 4, 5, 6\}$. All runs are performed to a final time of $t_{\text{final}} = 2.6$ and the CFL is kept fixed at 52. The reference solution is a S_{96}-S_{16} hybrid using a T_N quadrature in angle, a third-order DG method in space on a 448×448 grid, and a defect correction time integrator [10].

In Figure 3.9, we show densities of a few select runs, calculated using S_N and hybrid methods. The log of the corresponding relative L_2-errors are depicted in Figure 3.10. As before, the S_N solutions suffer from ray-effects that are marginally reduced as the number of angle increases. Meanwhile, most of the disparities between the reference and hybrid solutions can be attributed to stochastic noise. The hybrid has a mix of ray effects from the collided equation solve and particle tracks from the uncollided equation solve on the backside of the hohlraum However, these errors here are on the order of 10^{-2}-10^{-3}, which is much smaller than the errors in othe back of the domain.

Detailed comparisons between the relative error against the computational complexity are depicted in Figure 3.11. As before, increasing the angular resolution in the collided equation does not benefit the accuracy of the hybrid method. Increasing particles also has less effect than in the previous problems. The S_N solutions benefit most from finer spatial resolution, while the angular resolution does not matter as much. Spatial resolution also plays the biggest role for the hybrid. We do observe that for small particle counts (the purple points in the figure) increasing resolution can actually increase the error. This is explained by the fact that an under-sampled MC calculation does not benefit from more spatial resolution. Nevertheless, for a fixed spatial resolution, we do observe an improvement in the error when N_p is increased.

Overall the hybrid runs produce solutions with comparable or better accuracy than monolithic S_N runs with the same spatial resolution. Hybrid runs using S_8 for the collided equation achieve improved accuracy at nearly the same complexity while runs using S_4 for the collided equation achieved improved accuracy with even less complexity.

3.6 Conclusion and Discussion

In this work, we have presented a collision-based hybrid method that uses a Monte Carlo method for the uncollided solution and a discrete ordinate discretization for the collided solution. This combination of methods was originally proposed for the first collision source strategy used in [3] in the steady-state setting. Thus this work can be considered as an extension to the time-dependent setting that requires a remap procedure after every time step.

Experimental simulations have been performed on three standard benchmarks. For each benchmark, the results demonstrate that the hybrid method is more efficient, in the sense that it achieves greater accuracy with the comparable or less complexity or is less complexity with comparable or greater accuracy. Here complexity is a measure of how many unknowns are updated during particle moves for Monte Carlo or sweeping iterations for discrete ordinates.

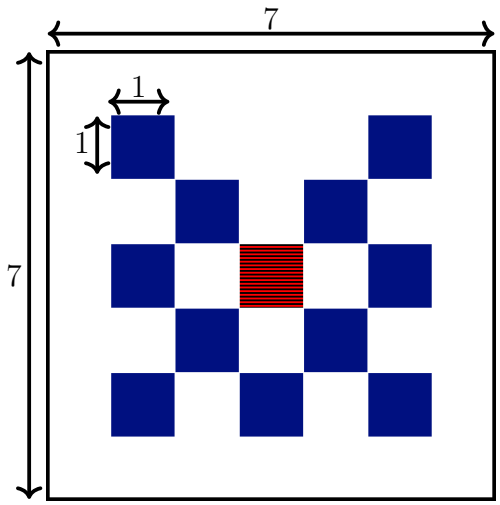
This work has concentrated on single-energy particle transport problems. However recent work has shown that when considering energy-dependent problems, more opportunities for hybridization arise when considering fully deterministic hybrids [42]. In those results it was shown that low-resolution in energy can be used for the collided solution as well as low-resolution in angle. With the introduction of Monte Carlo, new opportunities arise. For example, continuous energy cross-sections could be used in the uncollided portion. This could be important to treat resonances in neutron transport problems, but investigation is needed to quantify any benefits from this approach. The methodology here may also be extended to nonlinear RTE using standard linearization strategies, although corrections will need to be introduced to handle energy temperature dependent opacities. In addition future investigations should be made regarding the use of Monte Carlo techniques inside the high-order time accuracy methods developed for hybrid problems in [9, 10].

Finally, a clear strategy for choosing of discretization parameters does not exist at this point. The selection of spatial and temporal discretization parameters is much like any other method. However, the appropriate choice for the relative degrees of freedom in angle between the uncollided and collided equation is not clear; nor is the time interval to wait until the relabeling is performed. While some work exists to understand errors introduced by

the hybrid [16], the analysis is quite involved, even for a very simple case. Thus, the best approach is most likely an adaptive strategy based on a-posteriori estimates. This will be the topic of future work.

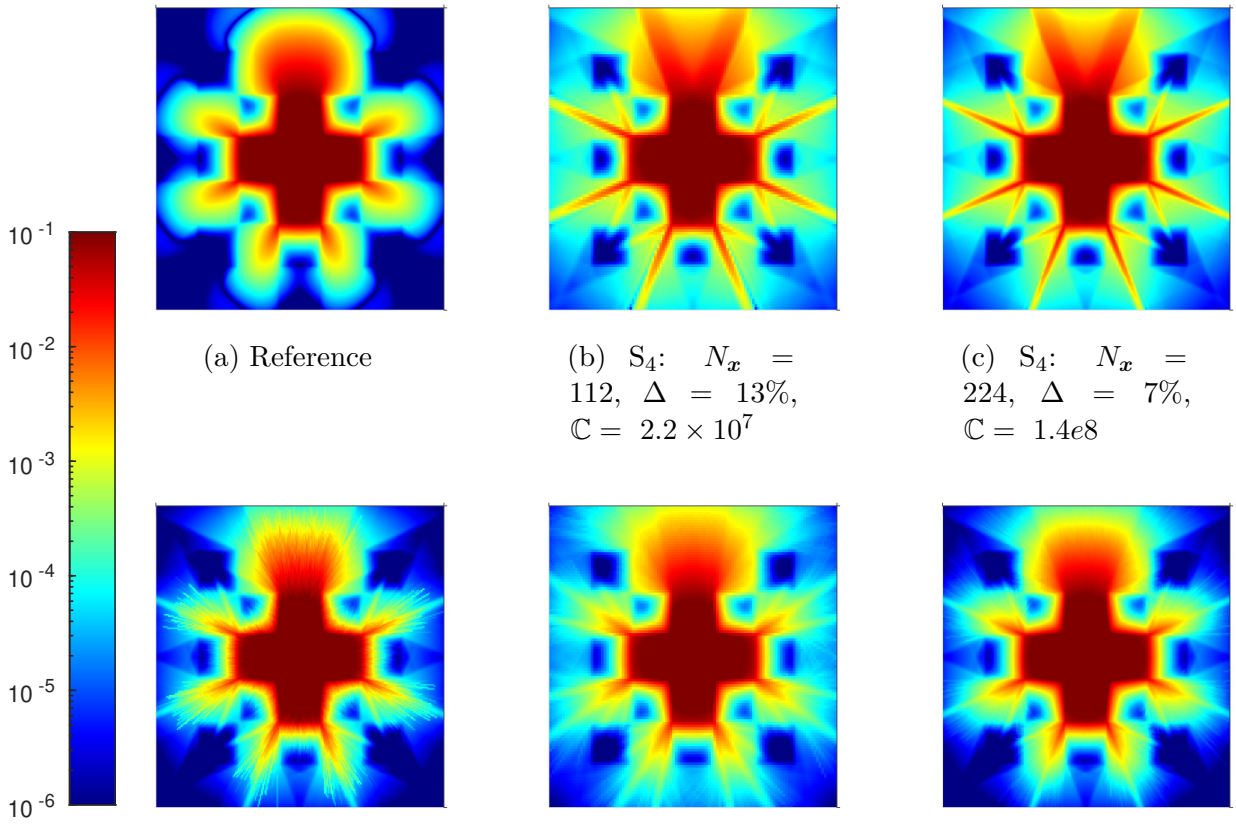
3.7 Acknowledgments

J.K. gratefully acknowledges support from the 2022 National Science Foundation Mathematical Sciences Graduate Internship to conduct this research at Oak Ridge National Laboratory.



	σ_a	σ_s	Q
blue(filled)	10	0	0
red(striped)	0	1	1
white	0	1	0

Figure 3.4: Geometric layout and table of material properties for the Lattice Problem. (In text mentions: p.75)



(a) Reference

(b) S_4: $N_{\boldsymbol{x}} = 112$, $\Delta = 13\%$, $\mathbb{C} = 2.2 \times 10^7$

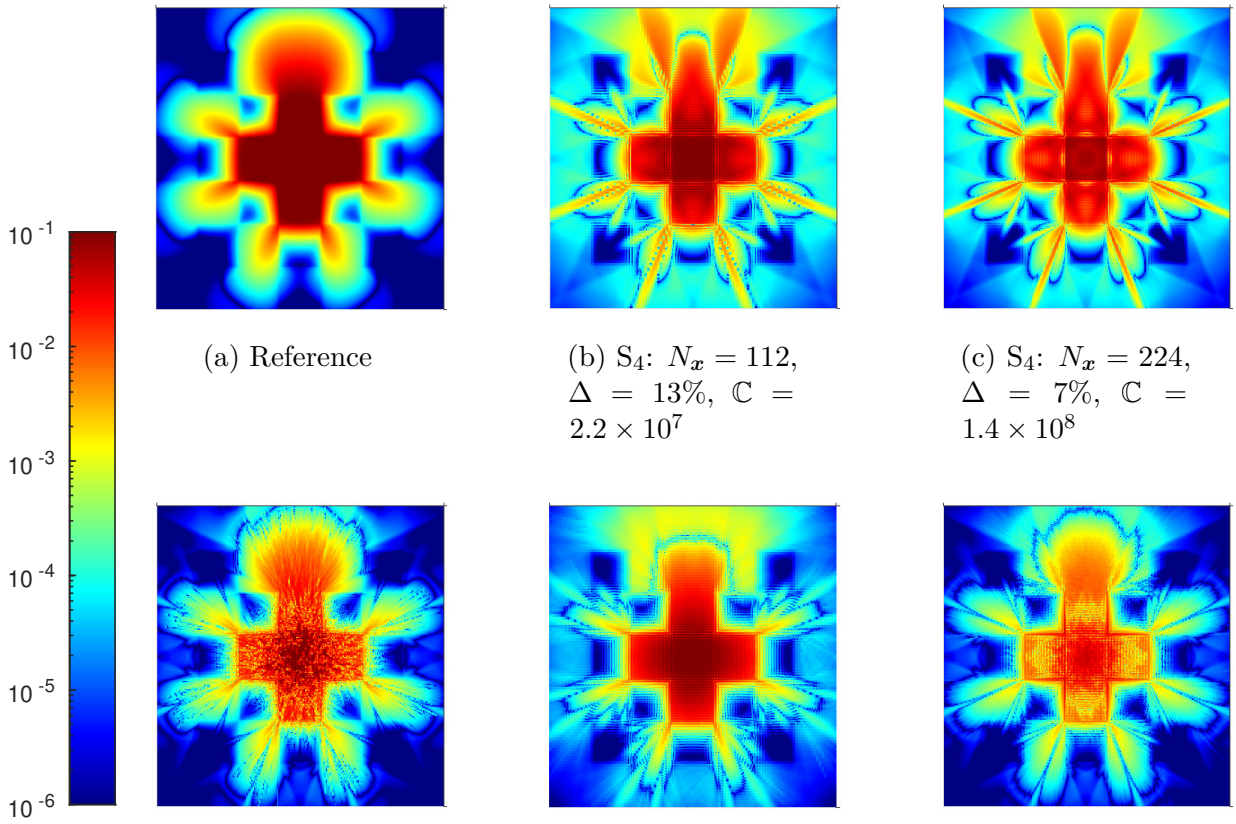
(c) S_4: $N_{\boldsymbol{x}} = 224$, $\Delta = 7\%$, $\mathbb{C} = 1.4e8$

(d) $H_{\text{MC-S}_4}$:$N_{\boldsymbol{x}} = 224$, $N_{\text{MC}}^{\text{tot}} = 1.2 \times 10^6$, $\Delta = 10\%$, $\mathbb{C} = 8.3 \times 10^7$

(e) $H_{\text{MC-S}_8}$: $N_{\boldsymbol{x}} = 112$, $N_{\text{MC}}^{\text{tot}} = 3.2 \times 10^6$, $\Delta = 13\%$, $\mathbb{C} = 1.8 \times 10^7$

(f) $H_{\text{MC-S}_4}$: $N_{\boldsymbol{x}} = 224$, $N_{\text{MC}}^{\text{tot}} = 1.3 \times 10^7$, $\Delta = 3\%$, $\mathbb{C} = 1.0 \times 10^8$

Figure 3.5: Numerical solutions to the lattice problem at $t = 3.2$ with CFL 25.6. Each numerical solution is characterized by a relative L^2 difference Δ with respect to the reference, defined in (3.48), and a complexity $\mathbb{C}$, defined in (3.49a) for the monolithic S_N method and (3.49b) for the hybrid. (In text mentions: p.79)



(a) Reference

(b) S_4: $N_{\boldsymbol{x}} = 112$, $\Delta = 13\%$, $\mathbb{C} = 2.2 \times 10^7$

(c) S_4: $N_{\boldsymbol{x}} = 224$, $\Delta = 7\%$, $\mathbb{C} = 1.4 \times 10^8$

(d) $H_{\text{MC-S}_4}$: $N_{\boldsymbol{x}} = 224$, $N_{\text{MC}}^{\text{tot}} = 1.2 \times 10^6$, $\Delta = 10\%$, $\mathbb{C} = 8.3 \times 10^7$

(e) $H_{\text{MC-S}_8}$: $N_{\boldsymbol{x}} = 112$, $N_{\text{MC}}^{\text{tot}} = 3.2 \times 10^6$, $\Delta = 13\%$, $\mathbb{C} = 1.8 \times 10^7$

(f) $H_{\text{MC-S}_4}$: $N_{\boldsymbol{x}} = 224$, $N_{\text{MC}}^{\text{tot}} = 1.3 \times 10^7$, $\Delta = 3\%$, $\mathbb{C} = 1.0 \times 10^8$

Figure 3.6: Lattice problem: Absolute difference between the reference solution and various numerical solutions at $t = 3.2$ with CFL 25.6. Each numerical solution is characterized by a relative L^2 difference Δ with respect to the reference, defined in (3.48), and a complexity $\mathbb{C}$, defined in (3.49a) for the monolithic S_N method and (3.49b) for the hybrid. (In text mentions: p.79)

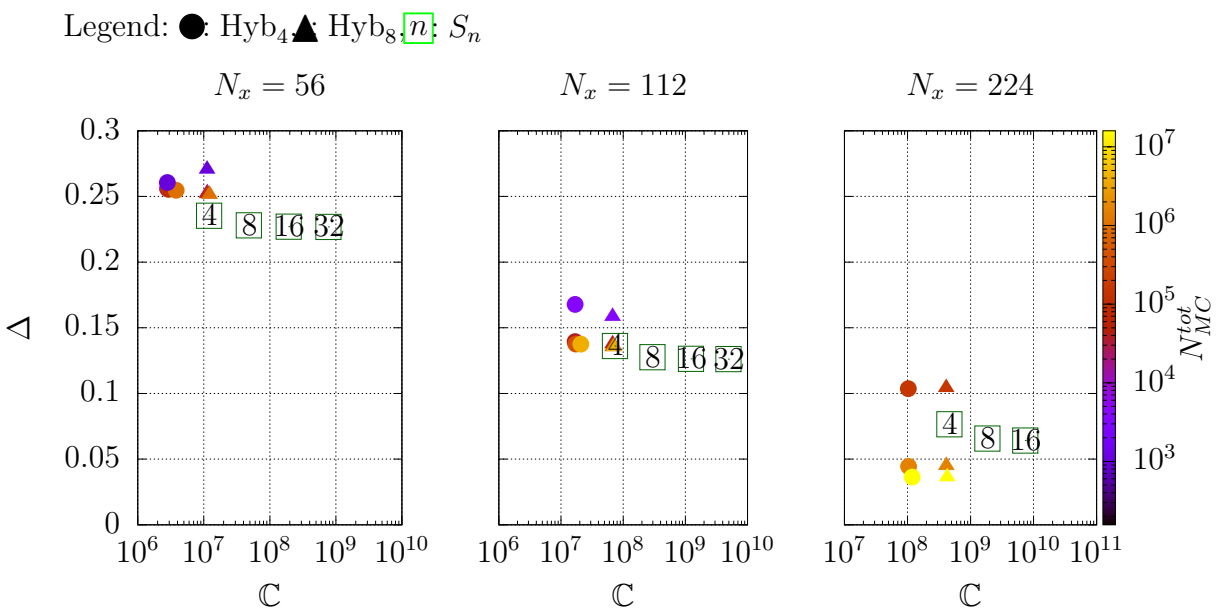
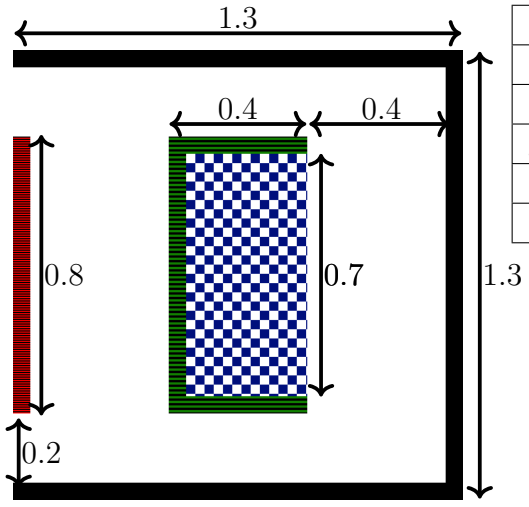
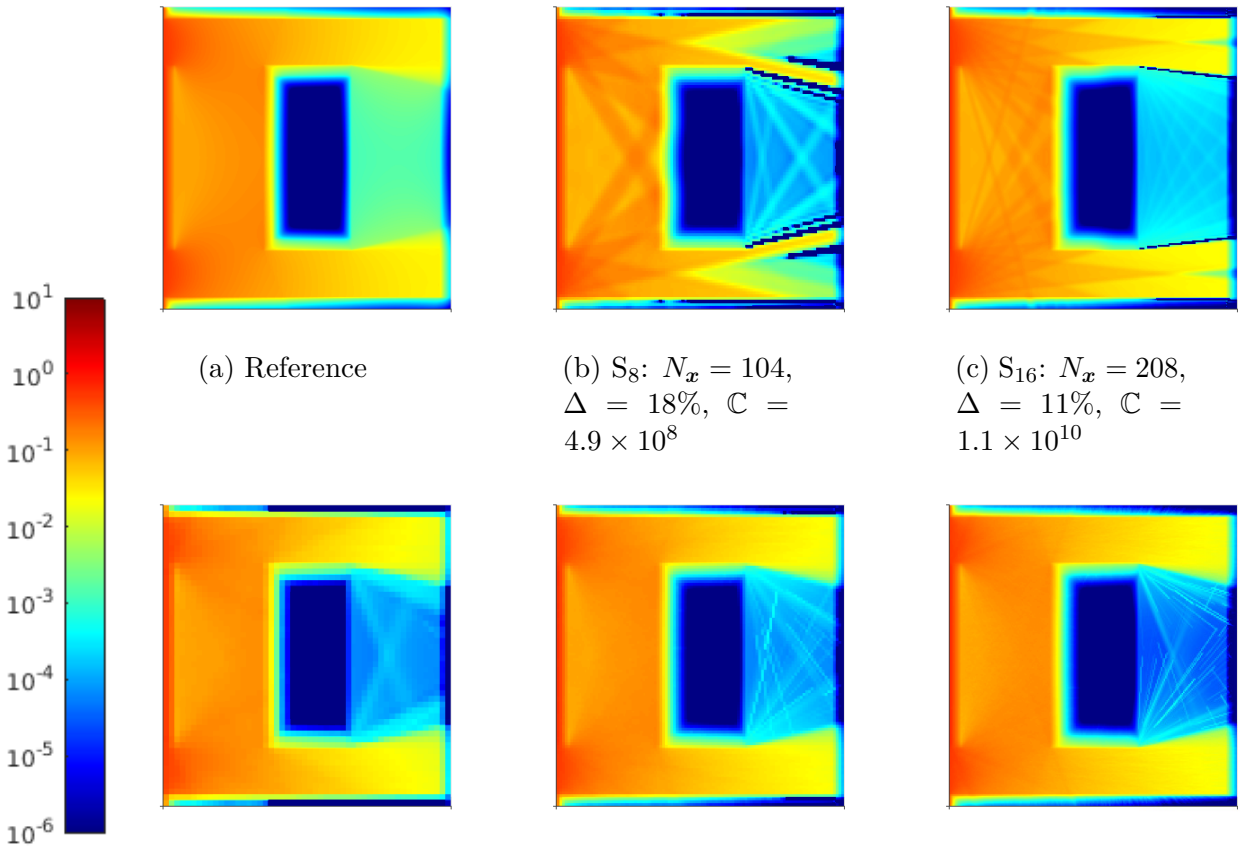


Figure 3.7: The relative L^2-difference Δ vs. complexity $\mathbb{C}$ for the scalar flux Φ in the lattice problem, using the hybrid method with S_4(filled circle markers), the hybrid with S_8 (filled triangle markers) and the monolithic S_N method (empty, green markers). Points that are down and to the left are more efficient. All methods where run for three different spatial resolutions: $N_x = 56$(left), $N_x = 112$(middle), $N_x = 224$(right). Coloring of the hybrid data points corresponds to the total number of particles N_{MC}^{tot} according to the colorbar. The S_N method was run for $N = 4, 8, 16, 32$. Each DG-data point is assigned a numerical label according to its value of N. The formula for Δ is given in (3.48) which the complexity $\mathbb{C}$ is given by (3.49a) for the monolithic S_N method and by (3.49b) for the hybrid. (In text mentions: pp.79,79,79)



	σ_a	σ_s	Q
black	0	100	0
red(thin stripes)	5	95	0
green (thick stripes)	10	90	0
blue (checkered)	50	50	0
white	0	0.1	0

Figure 3.8: Geometric layout and table of material parameters for the Hohlraum Problem. All walls have a thickness of 0.05. (In text mentions: p.79)



(a) Reference

(b) S_8: $N_{\boldsymbol{x}} = 104$, $\Delta = 18\%$, $\mathbb{C} = 4.9 \times 10^8$

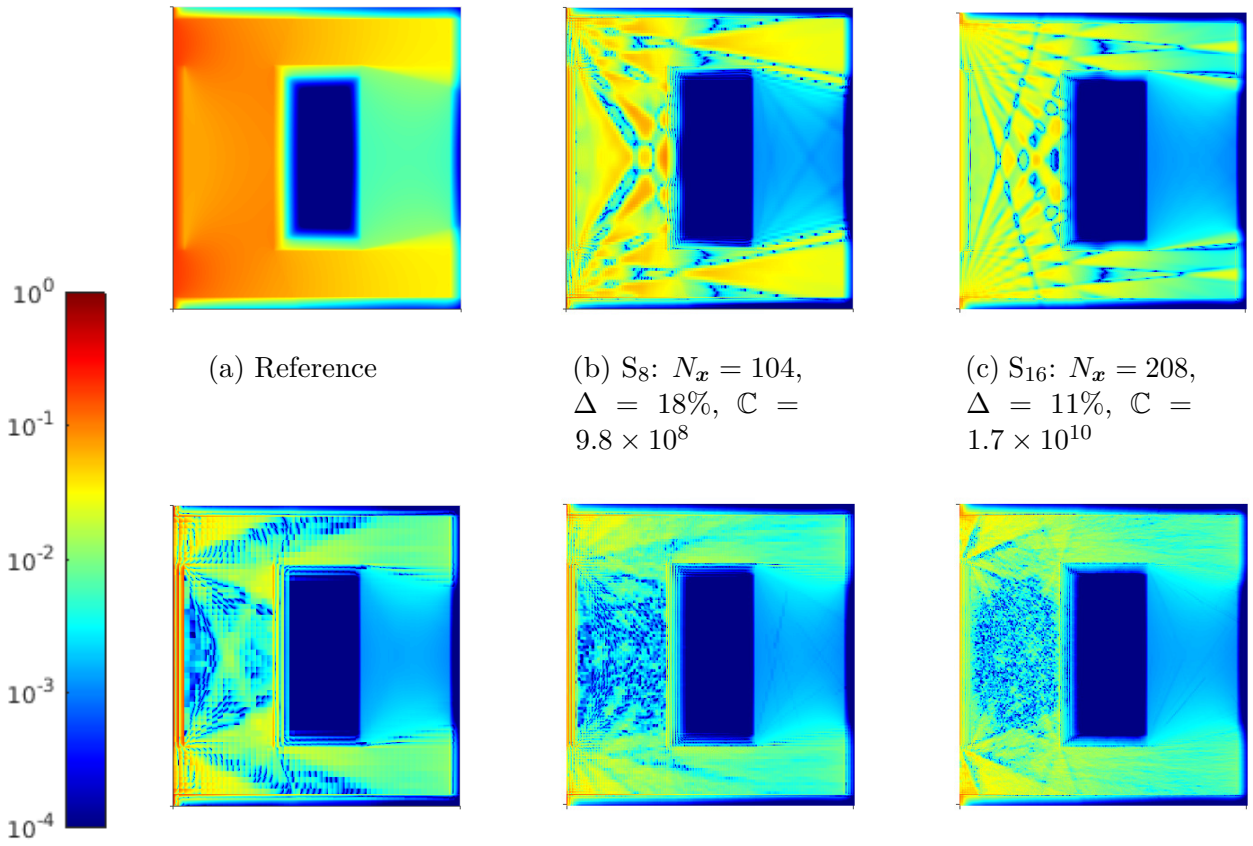
(c) S_{16}: $N_{\boldsymbol{x}} = 208$, $\Delta = 11\%$, $\mathbb{C} = 1.1 \times 10^{10}$

(d) $H_{\text{MC-S}_4}$:$N_{\boldsymbol{x}} = 52$, $N_{\text{MC}}^{\text{tot}} = 2.1e6$, $\Delta = 18\%$, $\mathbb{C} = 1.7 \times 10^7$

(e) $H_{\text{MC-S}_8}$:$N_{\boldsymbol{x}} = 104$, $N_{\text{MC}}^{\text{tot}} = 7.6 \times 10^6$, $\Delta = 11\%$, $\mathbb{C} = 9.7 \times 10^7$

(f) $H_{\text{MC-S}_4}$: $N_{\boldsymbol{x}} = 208$, $N_{\text{MC}}^{\text{tot}} = 2.9 \times 10^7$, $\Delta = 8\%$, $\mathbb{C} = 5.4 \times 10^8$

Figure 3.9: Numerical solutions to the hohlraum problem at $t = 2.6$ with CFL 52. Each numerical solution is characterized by a relative L^2 difference Δ with respect to the reference, defined in (3.48), and a complexity $\mathbb{C}$, defined in (3.49a) for the monolithic S_N method and (3.49b) for the hybrid. (In text mentions: p.80)



(a) Reference

(b) S_8: $N_{\boldsymbol{x}} = 104$, $\Delta = 18\%$, $\mathbb{C} = 9.8 \times 10^8$

(c) S_{16}: $N_{\boldsymbol{x}} = 208$, $\Delta = 11\%$, $\mathbb{C} = 1.7 \times 10^{10}$

(d) $H_{\text{MC-S}_4}$:$N_{\boldsymbol{x}} = 52$, $N_{\text{MC}}^{\text{tot}} = 2.1 \times 10^6$, $\Delta = 18\%$, $\mathbb{C} = 1.1 \times 10^7$

(e) $H_{\text{MC-S}_8}$:$N_{\boldsymbol{x}} = 104$, $N_{\text{MC}}^{\text{tot}} = 7.6 \times 10^6$, $\Delta = 11\%$, $\mathbb{C} = 9.7 \times 10^7$

(f) $H_{\text{MC-S}_4}$: $N_{\boldsymbol{x}} = 208$, $N_{\text{MC}}^{\text{tot}} = 2.9 \times 10^7$, $\Delta = 8\%$, $\mathbb{C} = 5.4 \times 10^8$

Figure 3.10: Hohlraum problem: Absolute difference between analytical solution to the line source problem and various numerical solutions at $t = 2.6$ with CFL 52. Each numerical solution is characterized by a relative L^2 difference Δ with respect to the reference, defined in (3.48), and a complexity $\mathbb{C}$, defined in (3.49a) for the monolithic S_N method and (3.49b) for the hybrid. (In text mentions: p.80)

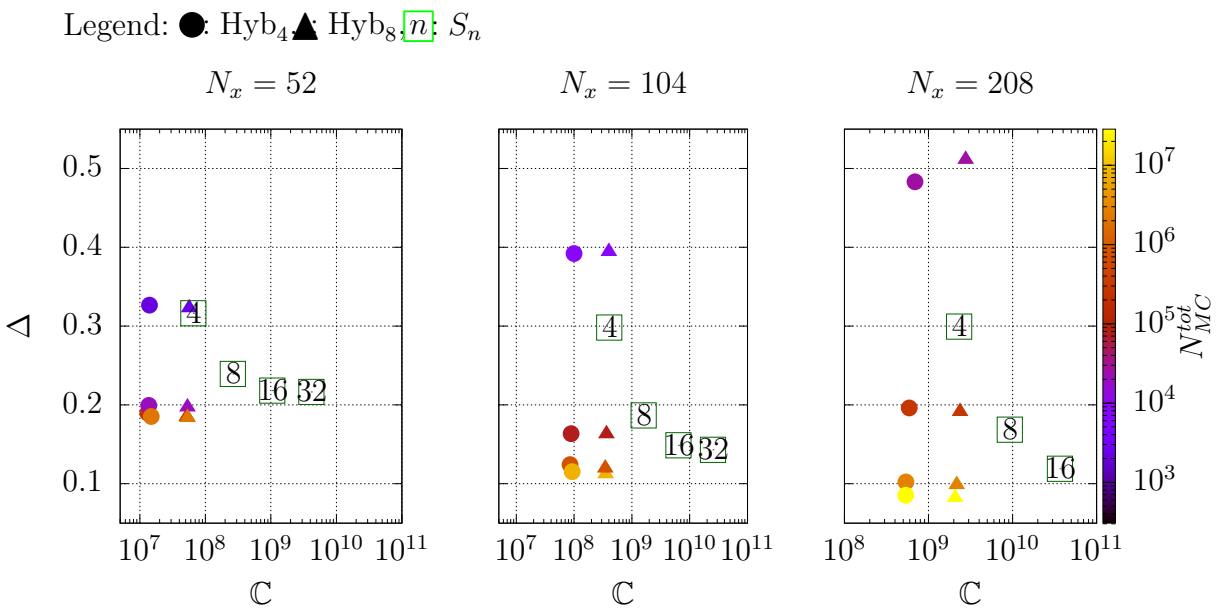


Figure 3.11: The relative L^2-difference Δ vs. complexity $\mathbb{C}$ for the scalar flux Φ in the hohlraum problem, using the hybrid method with S_4(filled circle markers), the hybrid with S_8 (filled triangle markers) and the monolithic S_N method (empty, green markers). Points that are down and to the left are more efficient. All methods where run for three different spatial resolutions $N_{\boldsymbol{x}} = 52$(left), $N_{\boldsymbol{x}} = 104$(middle), $N_{\boldsymbol{x}} = 208$(right). Coloring of the hybrid data points corresponds to the total number of particles N_{MC}^{tot} according to the colorbar. The S_N method was run for $N = 4, 8, 16, 32$. Each DG-data point is assigned a numerical label according to its value of N. The formula for Δ is given in (3.48) which the complexity $\mathbb{C}$ is given by (3.49a) for the monolithic S_N method and by (3.49b) for the hybrid. (In text mentions: p.80)

References

[1] Marvin L Adams. Discontinuous finite element transport solutions in thick diffusive problems. *Nuclear science and engineering*, 137(3):298–333, 2001. 60

[2] Timo Aila and Samuli Laine. Understanding the efficiency of ray traversal on gpus. In *Proceedings of the conference on high performance graphics 2009*, pages 145–149, 2009. 55

[3] Raymond E Alcouffe. A first collision source method for coupling monte carlo and discrete ordinates for localized source problems. In *Monte-Carlo Methods and Applications in Neutronics, Photonics and Statistical Physics: Proceedings of the Joint Los Alamos National Laboratory-Commissariat à l'Energie Atomique Meeting Held at Cadarache Castle, Provence, France April 22–26, 1985*, pages 352–366. Springer, 2006. 55, 57, 81

[4] Kendall Atkinson. Numerical integration on the sphere. *The ANZIAM Journal*, 23(3):332–347, 1982. 67

[5] Yousry Azmy, Enrico Sartori, Edward W Larsen, and Jim E Morel. Advances in discrete-ordinates methodology. *Nuclear computational science: A century in review*, pages 1–84, 2010. 59

[6] Guillaume Bal, Anthony B Davis, and Ian Langmore. A hybrid (monte carlo/deterministic) approach for multi-dimensional radiation transport. *Journal of Computational Physics*, 230(20):7723–7735, 2011. 55

[7] Troy L Becker, Allan B Wollaber, and Edward W Larsen. A hybrid monte carlo–deterministic method for global particle transport calculations. *Nuclear science and engineering*, 155(2):155–167, 2007. 55

[8] Thomas A Brunner. Forms of approximate radiation transport. Technical Report SAND2002-1778, Sandia National Laboratories, Albuquerque, NM (United States), 2002. 67, 79

[9] Michael M Crockatt, Andrew J Christlieb, C Kristopher Garrett, and Cory D Hauck. An arbitrary-order, fully implicit, hybrid kinetic solver for linear radiative transport using integral deferred correction. *Journal of Computational Physics*, 346:212–241, 2017. 55, 60, 67, 79, 81

[10] Michael M Crockatt, Andrew J Christlieb, C Kristopher Garrett, and Cory D Hauck. Hybrid methods for radiation transport using diagonally implicit runge–kutta and space–time discontinuous galerkin time integration. *Journal of Computational Physics*, 376:455–477, 2019. 55, 75, 80, 81

[11] Michael M Crockatt, Andrew J Christlieb, and Cory D Hauck. Improvements to a class of hybrid methods for radiation transport: Nyström reconstruction and defect correction methods. *Journal of Computational Physics*, 422:109765, 2020. 72

[12] Jeffery D Densmore and Edward W Larsen. Asymptotic equilibrium diffusion analysis of time-dependent Monte Carlo methods for grey radiative transfer. *Journal of Computational Physics*, 199(1):175–204, 2004. 54

[13] Jeffery D Densmore, Todd J Urbatsch, Thomas M Evans, and Michael W Buksas. A hybrid transport-diffusion method for monte carlo radiative-transfer simulations. *Journal of Computational Physics*, 222(2):485–503, 2007. 63

[14] Stephen A Dupree and Stanley K Fraley. *A Monte Carlo primer: A Practical approach to radiation transport*, volume 1. Springer Science & Business Media, 2002. 66

[15] Joseph A Fleck Jr. The calculation of nonlinear radiation transport by a monte carlo method. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 1961. 96

[16] Andrés Galindo-Olarte, Victor P DeCaria, and Cory D Hauck. Numerical analysis of a hybrid method for radiation transport. *arXiv preprint arXiv:2306.04714*, 2023. 82

[17] BD Ganapol. Homogeneous infinite media time-dependent analytic benchmarks for x-tm transport methods development. *Los Alamos National Laboratory*, 1999. 67, 72, 74

[18] C Kristopher Garrett and Cory D Hauck. A comparison of moment closures for linear kinetic transport equations: The line source benchmark. *Transport Theory and Statistical Physics*, 42(6-7):203–235, 2013. 72, 74

[19] Jean-Luc Guermond and Guido Kanschat. Asymptotic analysis of upwind discontinuous galerkin approximation of the radiative transport equation in the diffusive limit. *SIAM Journal on Numerical Analysis*, 48(1):53–78, 2010. 60

[20] Weimin Han, Jianguo Huang, and Joseph A Eichholz. Discrete-ordinate discontinuous galerkin methods for solving the radiative transfer equation. *SIAM Journal on Scientific Computing*, 32(2):477–497, 2010. 60

[21] Cory D Hauck and Ryan G McClarren. A collision-based hybrid method for time-dependent, linear, kinetic transport equations. *Multiscale Modeling & Simulation*, 11(4):1197–1227, 2013. 55, 56

[22] Vincent Heningburg and Cory D Hauck. Hybrid solver for the radiative transport equation using finite volume and discontinuous galerkin. *arXiv preprint arXiv:2002.02517*, 2020. 55

[23] Johannes Krotz, Cory D. Hauck, and Ryan G. McClarren. A hybrid monte carlo, discontinuous galerkin method for linear kinetic transport equations. 2023. 53

[24] Edward W Larsen, Jim E Morel, and Warren F Miller Jr. Asymptotic solutions of numerical transport problems in optically thick, diffusive regimes. *Journal of Computational Physics*, 69(2):283–324, 1987. 60

[25] Elmer Eugene Lewis and Warren F Miller. Computational methods of neutron transport. 1984. 54

[26] Ryan McClarren. *Computational nuclear engineering and radiological science using python*. Academic Press, 2017. 66, 67

[27] Ryan G McClarren and Todd J Urbatsch. A modified implicit monte carlo method for time-dependent radiative transfer with adaptive material coupling. *Journal of Computational Physics*, 228(16):5669–5686, 2009. 54

[28] Scott W Mosher and Farzad Rahnema. The incident flux response expansion method for heterogeneous coarse mesh transport problems. *Transport Theory and Statistical Physics*, 35(1-2):55–86, 2006. 54

[29] Madicken Munk and Rachel N Slaybaugh. Review of hybrid methods for deep-penetration neutron transport. *Nuclear Science and Engineering*, 2019. 55

[30] H Park, DA Knoll, RM Rauenzahn, AB Wollaber, and RB Lowrie. Moment-based acceleration of monte carlo solution for multifrequency thermal radiative transfer problems. *Journal of Computational and Theoretical Transport*, 43(1-7):314–335, 2014. 55

[31] Zhuogang Peng and Ryan G McClarren. A high-order/low-order (holo) algorithm for preserving conservation in time-dependent low-rank transport calculations. *Journal of Computational Physics*, 447:110672, 2021. 55

[32] Steve Pressé and Ioannis Sgouralis. *Data Modeling for the Sciences: Applications, Basics, Computations*. Cambridge University Press, 2023. 97

[33] Qiwei Sheng and Cory Hauck. Uniform convergence of an upwind discontinuous galerkin method for solving scaled discrete-ordinate radiative transfer equations with isotropic scattering. *Mathematics of Computation*, 90(332):2645–2669, 2021. 60

[34] Yi Shi, Xiaole Han, Wenjun Sun, and Peng Song. A continuous source tilting scheme for radiative transfer equations in implicit monte carlo. *Journal of Computational and Theoretical Transport*, 50(1):1–26, 2020. 54

[35] Jerome Spanier and Ely M Gelbard. *Monte Carlo principles and neutron transport problems*. Courier Corporation, 2008. 54

[36] Elad Steinberg and Shay I Heizler. Multi-frequency implicit semi-analog Monte-Carlo (ISMC) radiative transfer solver in two-dimensions (without teleportation). *Journal of Computational Physics*, 450:110806, 2022. 54

[37] CP Thurgood, A Pollard, and HA Becker. The tn quadrature set for the discrete ordinates method. 1995. 72

[38] John C Wagner and Alireza Haghighat. Automated variance reduction of Monte Carlo shielding calculations using the discrete ordinates adjoint function. *Nuclear Science and Engineering*, 128(2):186–208, 1998. 55

[39] John C Wagner, Douglas E Peplow, and Scott W Mosher. Fw-cadis method for global and regional variance reduction of monte carlo radiation transport calculations. *Nuclear Science and Engineering*, 176(1):37–57, 2014. 55

[40] Wallace F Walters. Use of the chebyshev-legendre quadrature set in discrete-ordinate codes. In *Proc. Conf. Numerical Methods in High Temperature Physics*, pages 2–6, 1987. 67

[41] Ben Whewell, Ryan G McClarren, Cory D Hauck, and Minwoo Shin. Multigroup neutron transport using a collision-based hybrid method. *Nuclear science and engineering*, pages 1–20, 2023. 55

[42] Ben Whewell, Ryan G McClarren, Cory D Hauck, and Minwoo Shin. Multigroup Neutron Transport Using a Collision-Based Hybrid Method. *Nuclear science and engineering*, 2023. 81

[43] Jeffrey Alan Willert. *Hybrid deterministic/Monte Carlo methods for solving the neutron transport equation and k-eigenvalue problem*. North Carolina State University, 2013. 55

[44] Allan B Wollaber. Four decades of implicit Monte Carlo. *Journal of Computational and Theoretical Transport*, 45(1-2):1–70, 2016. 54

[45] Dingkang Zhang and Farzad Rahnema. Comet solutions to a stylized bwr benchmark problem. Technical report, American Nuclear Society, Inc., 555 N. Kensington Avenue, La Grange Park ..., 2012. 54

Appendix: Boundary conditions for the Hohlraum problem

Unlike the line source and lattice problems, the linearized hohlraum problem involves non-zero boundary conditions. A Monte Carlo implementation of this boundary condition is stated in [15]; here we present a derivation of the approach that is used.

In the hohlraum problem, it is assumed that $\Psi(x, y, t, \Omega) = 1$ for $x = 0$ and $\Omega_x > 0$, i.e., a constant flux of 1 along the left boundary is assumed for each incoming direction. To model this with Monte Carlo, we assume that this flux is due to a source s_b (see (3.37c)) located on an infinitesimal slab just left of the boundary.

Consider first a finite slab $S_a = \{(x, y) \in [-a, 0] \times [0, 1.3]\}$, where $a > 0$. We assume that $\sigma_\mathrm{a} = \sigma_\mathrm{s} = 0$ on S, that the source $s_b(x, y, \mathbf{\Omega}; a) = s_b(x, \mathbf{\Omega}; a)$ is independent of y and t, and that

$$\hat{s}_b(\mathbf{\Omega}) := \int_{-a}^{0} s_b(x, \mathbf{\Omega}; a)dx \tag{3.51}$$

is independent of a. Thus, $s_b(\mathbf{x}, \mathbf{\Omega}, t) \to \hat{s}_b(\mathbf{\Omega})\delta(x)$ as $a \to 0$. To determine $\hat{s}_b$, we assume that Ψ is independent of y and t on S and satisfies the steady-state equation

$$\Omega_x \Psi_x(x, y, \mathbf{\Omega}) = s_b, \qquad (x, y) \in S, \quad \mathbf{\Omega} \in \mathbb{S}^2, \tag{3.52a}$$

$$\Psi(-a, y, \mathbf{\Omega}) = 0, \qquad y \in [0, 1.3], \quad \Omega_x > 0. \tag{3.52b}$$

This formulation is consistent with (3.34), given the assumptions made on Ψ, s_b, and the material cross-sections. Integrating (3.52b) with respect to x and applying the boundary condition in (3.52b) gives

$$\hat{s}_b(\mathbf{\Omega}) = \Omega_x, \qquad \Omega_x > 0. \tag{3.53}$$

Hence $s_b = \Omega_x \delta(x)$.

Finally, to sample particles according to the probability density $\rho(\Omega_x) \propto \hat{s}_b$ for $\Omega_x > 0$, we compute the cumulative distribution function (CDF):

$$F(\Omega_x) = \int_0^{\Omega_x} 2\mu d\mu = \Omega_x^2. \tag{3.54}$$

According to the fundamental theorem of simulation [32, pp. 19-22], the correct angular distribution can be sampled by generating uniform variables $u \in [0, 1]$ and setting $\Omega_x = F^{-1}(u) = \sqrt{u}$.

Chapter 4

A Likelihood Approach to Filtering for Advection Diffusion Processes

4.1 Disclosure

This chapter is, up to formatting, identical to the manuscript of the same name [8]. The manuscript is collaborative work with Jorge. M. Ramirez and Juan M. Restrepo and at the time this document is written under review in the journal *Monthly Weather Review*. The ideas and results presented build on previous work by Juan M. Restrepo. New ideas and results presented were worked out by Johannes Krotz under guidance of Juan M. Restrepo and Jorge M. Ramirez. The manuscript was cowritten with Juan M. Restrepo and Jorge Ramirez.

4.2 Abstract

A Bayesian data assimilation scheme is formulated for advection-dominated advective and diffusive evolutionary problems, based upon the Dynamic Likelihood (DLF) approach to filtering. The DLF was developed specifically for hyperbolic problems –waves–, and in this paper, it is extended via a split step formulation, to handle advection-diffusion problems. In the dynamic likelihood approach, observations and their statistics are used to propagate probabilities along characteristics, evolving the likelihood in time. The estimate posterior thus inherits phase information. For advection-diffusion the advective part of the time evolution is handled on the basis of observations alone, while the diffusive part is informed through the model as well as observations. We expect, and indeed show here, that in advection-dominated problems, the DLF approach produces better estimates than other assimilation approaches, particularly when the observations are sparse and have low uncertainty. The added computational expense of the method is cubic in the total number of observations over time, which is on the same order of magnitude as a standard Kalman filter and can be mitigated by bounding the number of forward propagated observations, discarding the least informative data.

4.3 Introduction

A general framework in Bayesian estimation to assimilate observations and model predictions has become known as data assimilation. Models are used to inform a prior and observations inform the likelihood. For time-dependent problems, the estimation objective is to find the evolution of moments of the posterior of a time-dependent state variable, conditioned on observations. A variety of computational methodologies have been proposed to accomplish this (see [17] and references therein). In linear problems with Gaussian noise processes, the variance minimizer estimate of the time-dependent mean and variance of the posterior can be obtained sequentially by the Kalman Smoother [12] or partially by the Kalman Filter (KF) [7]. Kushner, Stratanovich, Pardoux (see, for example, [9]) proposed a variance minimizer estimate for the nonlinear/non-Gaussian problem, however, it is computationally tractable only for very low-dimensional state variable problems. Successful approximations of the estimate can sometimes be obtained via generalizations like the Extended Kalman Filter [10, 18], or the Unscented Kalman Filter [6], among others. Sample estimates can be approximated via the Ensemble Kalman Filter [3, 4] and its variants, the path integral method [1], and various particle filter schemes [11, 2, 14]. There are estimators that have special properties (see [15]) or that exploit the underlying dynamics of the problem. An example of the latter is the the dynamic likelihood filtering approach (DLF), first proposed in [13].

The DLF is denoted an "approach" rather than a filtering method because, in principle, it applies to any of the linear or nonlinear data assimilation methodologies. It was developed specifically for problems in wave dynamics, in general, hyperbolic partial differential equations. The crux of the DLF approach is to modify the conditional, posterior distribution of the state variable by exploiting a property peculiar to wave problems: finite-time propagation of information, which is utilised to propose a dynamic likelihood. A second aspect of DLF is that it tracks the state variables of the partial differential equation along characteristics, thus obtaining stochastic differential equations. Peculiarities of the DLF approach are that phase information enters directly into the estimation, and that we can make Bayesian estimates at times when observations are available and when they are not (even in the near future).

The main practical advantage of the method is that it addresses the more common situation in wave problems: sparse observation networks that are, nevertheless, fairly low in noise. Under these circumstances, as was shown in [5], the DLF produces superior estimates when compared to the best traditional estimator.

In this paper, we develop the dynamic likelihood approach for data assimilation problems in transport modeled by forced advection-diffusion equations. We thus expand the range of applicability of this estimation approach to an important class of dynamics. We will focus on finding estimates of quantities of interest when the source of uncertainties appears in the advection process and the forcing. The statement of the problem appears in Section 4.4. The DLF evolves the likelihood forward in time along characteristics by generating *pseudo-observations* at times between actual observations. A pseudo-observation is derived from a real observation at a previous time. The pseudo-observation framework appears in Section 4.5.1. This section also details how the DLF approach applies to the advection-diffusion dynamics, using techniques and ideas similar to a Kalman Filter.

To appreciate the practicality of the methodology, we present in Section 4.6 an accounting of the cost of implementing the DLF on a sequential estimator data assimilation method. In Section 4.7.2, we compare the DLF approach proposed for advection-diffusion problems to the outcomes obtained via a Kalman Filter because the Kalman estimates for this problem are familiar, optimal, and easily understood. A discussion and conclusions appear in Section 4.8.

4.4 Statement of the Problem

At issue is the estimation of the posterior covariance of a noisy scalar state variable $u(x,t)$ given noisy observations, and the minimization of its trace. Here and throughout, x denotes space and t, time, The state variable obeys a noisy advection-diffusion initial value problem. We will develop a DLF approach to a particular filtering estimation scheme. We focus on linear dynamics since it allows us to evaluate the DLF approach in comparison with optimal filtering schemes, nevertheless, we argue that the development presented herein will extend to a number of nonlinear cases.

We are motivated to consider the DLF approach to the dynamics of advection-diffusion because it was shown in [5] that for hyperbolic dynamics, the DLF returned significantly better estimates on noisy hyperbolic problems, particularly when the observations were sparse yet had low uncertainty –which is the more common practical situation. We will, in fact, show that for the advection-dominated case, the DLF approach yields better estimates than other filtering approaches.

Since we are specializing to the linear advection-diffusion initial value problem with known Gaussian noise processes it is possible to fully determine the posterior distribution with the determination of the posterior mean and variance. We connote a sample time series from the distribution of $u(x,t)$ as the **truth**. We will make use of the truth for testing the performance of the DLF. In practice, the truth is not available to us. Instead, we are given an approximate solution of the stochastic advection-diffusion initial value problem, with known errors, often in the form of a computer code. The estimation problem will thus be one of finding moments of the posterior model state variable, given observations.

4.4.1 Dynamics, Model, Observations

The space interval over which the dynamical system is defined will be $[0,L] \subset \mathbb{R}$, with periodic boundaries. Space will be discretized by a grid X of equidistant nodes $X = \left\{x^k = k \cdot \Delta x\right\}_{k=0}^{K}$, with $x^K = L - \Delta x$ due to the periodic boundary conditions. The time interval shall be $[0,t_N] \subset \mathbb{R}$ discretized in equal time steps $T := \{t_n = n\Delta t\}_{n=0}^{N}$. We will denote the set T as *estimation times*. On $[0,L] \times [0,t_N]$ we consider the random field $u(x,t)$, which obeys the stochastic initial value problem

$$\begin{aligned} u_t - C(x,t)u_x &= Du_{xx} + F(x,t), \quad t > 0, \quad x \in [0,L], \\ u(x,0) &= u_0(x), \quad x \in [0,L]. \end{aligned} \tag{4.1}$$

The subscripts x and t connote partial differentiation with respect to these variables. The periodic initial condition is $u_0(x)$ is known or is drawn from an assumed known probability distribution $\mathcal{U}$. The parameter D is the diffusion constant, $F(x,t)$ and $C(x,t)$ are forcing

and wave speed terms, respectively. It will be assumed that

$$C(x,t) = c(x,t) + \phi(x,t), \tag{4.2}$$

$$F(x,t) = f(x,t) + \chi(x,t), \tag{4.3}$$

where $f(x,t)$ and $c(x,t)$ are the forcing and phase speed, respectively, which are assumed known, deterministic and periodic on $[0,L]$. The random fields ϕ and χ have the form $\phi(x,t)dt = AdW^c(x,t)$, $\chi(x,t)dt = BdW^u(x,t)$ with A and B known constants, dW^c and dW^u incremental zero-mean Wiener processes, assumed uncorrelated. We define a semi-continuous ensemble member solution to (4.1) on the space grid as $\boldsymbol{U}(t) := \left(U^k(t)\right)_{k=0}^K = (u(X,t))_{k=0}^K$.

Going forward, bold variables will denote vectors or matrices. Superindices are space, subindices are time. For all variables with a single index, e.g. a_i, we denote by $a_{k:n} = \{a_i\}_{i=k}^n$ the union over all indices between k and n.

We connote v as the **model** approximation to (4.1). We will build a specific one here as follows: on the grid $X \times T$, the values of v are obtained by a forward numerical solution of the SDE (4.1). At each time t_n, we denote the collection of values of the model $v(\cdot, t_n)$ on X by

$$\boldsymbol{V}_n = v(X, t_n) \tag{4.4}$$

The vector $\boldsymbol{V}_n$ is evolved forward with the SDE solver

$$\boldsymbol{V}_{n+1} = \boldsymbol{L}_n \boldsymbol{V}_n + \sqrt{\Delta t}\ \Delta \boldsymbol{w}_n + \Delta t\ \boldsymbol{f}_n \tag{4.5}$$

where $\boldsymbol{L}_n \in \mathbb{R}^{K\times K}$ is a numerical operator approximating the linear terms in (4.1), $\boldsymbol{f}_n := \left(f(x^k, t_n)\right)_{k=0}^K$ and $\Delta \boldsymbol{w}_n \in \mathbb{R}^N$ is mean-zero Gaussian vector with covariance matrix $\boldsymbol{Q}_n$ accounting for the stochasticity of (4.1) and the model error. The distribution of $\boldsymbol{V}_0$ and $\langle \boldsymbol{w}_n \boldsymbol{w}_n{}^\top \rangle$ are assumed known. For x-values that are off-grid, we use *linear interpolation in space* to extend the outputs of the numerical SDE solver (4.5) to $[0,L] \times T$. Namely, for $x \notin X$, we define

$$v(x, t_n) = \boldsymbol{H}(x) \boldsymbol{V}_n, \quad n = 1, \ldots, N \tag{4.6}$$

where $\boldsymbol{H}$ is a linear interpolation operator to be specified later. The model, up to time t_{n_m}, is used to inform the prior $\pi(\boldsymbol{V}_{0:n_m})$.

In practice, observations are obtained from instruments and the error is instrument-related. Here we generate them synthetically from the truth. We assume that **observations** are available at observation times $\{t_{n_1}, \ldots, t_{n_M}\} =: T_O \subset T$. The set of available observations is $\mathcal{O} := \{(\boldsymbol{y}_m, \boldsymbol{Y}_m)\}_{m=1}^M$ which provide, up to noise, temporally and spatially localized records on the value of u. Specifically, the observation pair $(\boldsymbol{y}_m, \boldsymbol{Y}_m)$ corresponds to a time t_{n_m} in the set of observation times $T_O \subset T$. The vector $\boldsymbol{y}_m \in [0, L]^I$ contains the $I \in \mathbb{N}$ locations where the observations were recorded, and $\boldsymbol{Y}_m \in \mathbb{R}^I$ is a measurement of the value of u at those locations at time t_{n_m}. Specifically,

$$Y_m^i = u(y_m^i, t_{n_m}) + \epsilon_m^i, \quad i = 1, \ldots, I \tag{4.7}$$

where the measurement error $\boldsymbol{\epsilon}_m$ is a mean-zero, normal vector in $\mathbb{R}^I$ with known covariance. Note that observation times $T_O \subset T$ do not include all times in T and that the number of observations I does not depend on t.

By Bayes Law

$$\pi(\boldsymbol{V}_{0:n_m}|\boldsymbol{Y}_{1:m}) \propto \pi(\boldsymbol{Y}_{1:m}|\boldsymbol{V}_{0:n_m})\pi(\boldsymbol{V}_{0:n_m}). \tag{4.8}$$

The likelihood at time $t_{n_m} \in T_O$, informed by observations is $\pi(\boldsymbol{Y}_{1:m}|\boldsymbol{V}_{0:n_m})$. The prior $\pi(\boldsymbol{V}_{0:n_m})$ is informed by the model.

4.4.2 The Kalman Filter (KF)

In principle, the DLF approach applies to most sequential filtering schemes. Since the problem we are considering is linear, we will be testing the DLF approach to filtering as applied to the Kalman filter (KF). In what follows it will be understood that a comparison between the DLF approach and the Kalman filter is to be understood as the DLF approach applied to the Kalman filter and the classical Kalman filter.

Referring to (4.8) the posterior at any time $t_{n_m} \in T_O$ can be split up as follows

$$\pi(\boldsymbol{V}_{0:n_m}|\boldsymbol{Y}_{1:m}) \propto \pi(\boldsymbol{Y}_m|\boldsymbol{V}_{n_m})\pi(\boldsymbol{V}_{0:n_m}|\boldsymbol{Y}_{1:m-1}) \tag{4.9}$$

$$\pi(\boldsymbol{V}_{0:n_m}|\boldsymbol{Y}_{1:m-1}) = \pi(\boldsymbol{V}_{n_m}|\boldsymbol{V}_{n_m-1})\pi(\boldsymbol{V}_{0:n_m-1}|\boldsymbol{Y}_{1:m-1}). \tag{4.10}$$

Let $\boldsymbol{V}_{n|n}$, denote the posterior of $\boldsymbol{V}_n$ conditioned on observations up to t_n. If $t_n = t_{n_m} \in T_O$ for some m, then this distribution is simply $\pi(\boldsymbol{V}_{n_m}|\boldsymbol{Y}_{1:m})$, which by (4.9) can be written as

$$\begin{aligned}
\boldsymbol{V}_{n_m|n_m} &\sim \int \pi(\boldsymbol{V}_{0:n_m}|\boldsymbol{Y}_{1:m}) d\boldsymbol{V}_{n_m-1} \cdots d\boldsymbol{V}_0 \\
&\propto \pi(\boldsymbol{Y}_m|\boldsymbol{V}_{n_m}) \int \pi(\boldsymbol{V}_{0:n_m-1}|\boldsymbol{Y}_{1:m-1}) d\boldsymbol{V}_{n_m-1} \cdots d\boldsymbol{V}_0 \\
&= \pi(\boldsymbol{Y}_m|\boldsymbol{V}_{n_m})\pi(\boldsymbol{V}_{n_m}|\boldsymbol{V}_{n_m-1|n_m-1}).
\end{aligned} \tag{4.11}$$

This prior $\pi(\boldsymbol{V}_{n_m}|\boldsymbol{V}_{n_m-1|n_m-1})$ can be calculated by applying the forward SDE solver (4.5) to $\boldsymbol{V}_{n-1|n-1}$, while the Likelihood $\pi(\boldsymbol{Y}_m|\boldsymbol{V}_{n_m})$ is determined entirely through the measurement errors. If, on the other hand, $t_n \notin T_O$, we simply have $\boldsymbol{V}_{n|n} \sim \pi(\boldsymbol{V}_n|\boldsymbol{V}_{n-1|n-1})$, which can be calculated through the model. Thus the posterior of $\boldsymbol{V}_{0:N}$ can be calculated sequentially for one $\boldsymbol{V}_n$ at a time, based on the posteriors up to the respective previous time step. All priors and likelihoods here, and therefore the posteriors too, are normally distributed.

We denote by $\langle \boldsymbol{V}_{n|n} \rangle$ and $\boldsymbol{P}_{n|n}$ the mean and covariance of $\boldsymbol{V}_{n|n}$ and by $\langle \boldsymbol{V}_{n|n-1} \rangle$ and $\boldsymbol{P}_{n|n-1}$ the mean and covariance of $\boldsymbol{V}_{n|n-1} \sim \pi(\boldsymbol{V}_n|\boldsymbol{V}_{n-1|n-1})$. Further, let $\boldsymbol{R}_m = \langle \boldsymbol{\epsilon}_m \boldsymbol{\epsilon}_m^\top \rangle$.

The KF produces sequential estimates for $\langle \boldsymbol{V}_{n|n} \rangle$ and $\boldsymbol{P}_{n|n}$, and thus for the posterior distribution, in two steps. In the *forecast* step the model is used to produce an initial estimate of $\pi(\boldsymbol{V}_n|\boldsymbol{V}_{n-1|n-1})$. Since the model is linear and the noise is (unbiased) normal, the prior at t_n is estimated through the model and the posterior at the previous time step:

$$\langle \boldsymbol{V}_{n|n-1} \rangle = \boldsymbol{L}_{n-1} \langle \boldsymbol{V}_{n-1|n-1} \rangle + \Delta t \boldsymbol{f}_{n-1}, \quad n = 1, \ldots, N, \tag{4.12}$$

$$\boldsymbol{P}_{n|n-1} = \boldsymbol{L}_{n-1} \boldsymbol{P}_{n-1|n-1} \boldsymbol{L}_{n-1}^\top + \boldsymbol{Q}_{n-1}, \quad n = 1, \ldots, N, \tag{4.13}$$

with $\boldsymbol{Q}_{n-1} = \langle \boldsymbol{w}_{n-1} \boldsymbol{w}_{n-1}^\top \rangle$. Initial data is assumed to be known or a sample of a known distribution. If no observations are available at time t_n, the posterior is not affected by the

likelihood, and thus $\langle \boldsymbol{V}_{n|n} \rangle = \langle \boldsymbol{V}_{n|n-1} \rangle$, and $\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1}$. If, on the other hand, observations are available, i.e. $n = n_m$ with $t_n = t_{n_m} \in T_O$, an *analysis* step is performed, which takes in the mean and covariance of the prior $\langle \boldsymbol{V}_{n|n-1} \rangle$ and $\boldsymbol{P}_{n|n-1}$ and the mean and covariance of the likelihood $\langle \boldsymbol{Y}_n \rangle$ and $\boldsymbol{R}_n$ and calculates the moments of the posterior. For any step t_{n_m} with observations, thus, the analysis step consists of the update

$$\langle \boldsymbol{V}_{n_m|n_m} \rangle = \langle \boldsymbol{V}_{n_m|n_m-1} \rangle + \boldsymbol{K}_{n_m} \left(\langle \boldsymbol{Y}_m \rangle - \boldsymbol{H}(\boldsymbol{y}_m) \langle \boldsymbol{V}_{n_m|n_m-1} \rangle \right), \tag{4.14}$$

$$\boldsymbol{P}_{n_m|n_m} = (\boldsymbol{I} - \boldsymbol{K}_{n_m} \boldsymbol{H}(\boldsymbol{y}_m)) \boldsymbol{P}_{n_m|n_m-1}. \tag{4.15}$$

Here $\boldsymbol{H}(\cdot)$ evaluated at the vector $\boldsymbol{v} \in \Omega^I$ is the interpolation matrix defined as $\boldsymbol{H}(\boldsymbol{v}) := (\boldsymbol{H}(v^i))_{i=1}^I \in \mathbb{R}^{I \times K}$. The term $\left(\langle \boldsymbol{Y}_m \rangle - \boldsymbol{H}(\boldsymbol{y}_m) \langle \boldsymbol{V}_{n_m|n_m-1} \rangle \right)$ in (4.14) is called the *innovation*. In (4.15) $\boldsymbol{I}$ is the N-dimensional identity matrix; $\boldsymbol{K}_{n_m}$ is called the Kalman gain and is defined as

$$\begin{aligned} \boldsymbol{K}_{n_m} =& \boldsymbol{P}_{n_m|n_m-1} \boldsymbol{H}(\boldsymbol{y}_m)^\top \\ & \cdot \left[\boldsymbol{H}(\boldsymbol{y}_m) \boldsymbol{P}_{n_m|n_m-1} \boldsymbol{H}(\boldsymbol{y}_m)^\top + \boldsymbol{R}_m \right]^{-1}. \end{aligned} \tag{4.16}$$

4.5 The Dynamic Likelihood Approach

In hyperbolic dynamics, (waves) information (along with uncertainties) will flow along characteristics. The DLF approach exploits the wave dynamics to propose a richer likelihood than other traditional problems. An observation $(\boldsymbol{y}_m, \boldsymbol{Y}_m)$ measured at time $t_{n_m} \in T_O$ is used to generate *pseudo-observations* $(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m)$, at times $t_n \in T$ with $t_n > t_{n_m}$. These are used to generate likelihood distributions in between observations. In fact, it is possible to produce likelihoods in the future, so that in principle, it is possible to do Bayesian estimation *in the future*. We refer to a likelihood, constructed from observations as well as from pseudo-observations, as the *dynamic likelihood*. The pseudo-observations are tightly coupled to the inherent model dynamics; in [5] we show how these are constructed for hyperbolic problems. Here we show how these could be formulated for advection-diffusion dynamics.

4.5.1 Pseudo-Observations

The pseudo-observation at time t_n, from the observation $\boldsymbol{y}_m$ at time t_{n_m}, have locations $\boldsymbol{\mathcal{H}}_n\boldsymbol{y}_m = \boldsymbol{x}_m(t_n) \in [0, L]^I$ which solve

$$\begin{aligned} dx_m^i(t) &= -c(x_m^i(t), t)dt, & t_{n_m} < t \leq t_N, \\ x_m^i(t_{n_m}) &= y_m^i & i = 1, \ldots, I \end{aligned} \tag{4.17}$$

A schematic of how $\boldsymbol{\mathcal{H}}_n\boldsymbol{y}_m$ arises from $\boldsymbol{y}_m$ is depicted in Figure 4.1. The value $\boldsymbol{\mathcal{H}}_n\boldsymbol{Y}_m$ of the pseudo-observations approximate u at the location of the characteristics, and they include a measurement error. Namely, $\boldsymbol{\mathcal{H}}_n\boldsymbol{Y}_m = (v(x_m^i(t_n), t_n) + \zeta^i(t_n))_{i=1}^I$, where $\boldsymbol{\zeta}^i(t_m)$ has the same distribution as ϵ_m^i.

Ideally, $\boldsymbol{\mathcal{H}}_n\boldsymbol{Y}_m$ would be the values at $t = t_n$ of solutions $u^i(t) := u(x_m^i(t), t)$ for $t_{n_m} < t \leq t_N$ of the following system of equations derived from (4.1) and (4.17),

$$\begin{aligned} du^i(t) =& \left((D + \frac{1}{2}A^2)u_{xx}^i(t) + f(x_m^i(t), t)\right)dt \\ &+ BdW^u + Au_x^i(t)dW^c \\ u^i(t_{n_m}) =& Y_m^i. \end{aligned} \tag{4.18}$$

A schematic of how a single pseudo-observation $\boldsymbol{\mathcal{H}}_n\boldsymbol{Y}_m^i$ would behave along a characteristic is depicted in Figure 4.2.

As the truth u is assumed unknown, we recast equation (4.18) with an approximate form where the terms u_x^i and u_{xx}^i are replaced by approximations $v_{(x)}(x,t) \approx u_x(x,t)$ and $v_{(xx)}(x,t) \approx u_{xx}(x,t)$ which we refer to as first and second derivatives of the model. (We could obtain $v_{(x)}(x, t_n)$ and $v_{(xx)}(x, t_n)$ from the interpolated model v, this is not a requirement and they could just as well be given from another model or data). With these replacements (4.18) becomes

$$\begin{aligned} d\tilde{u}^i(t) =& \left(\left(D + \frac{A^2}{2}\right)v_{(xx)}(x_m^i(t), t) + f\right)dt \\ &+ Av_{(x)}((x_m^i(t), t))dW^c + BdW^u \end{aligned}$$

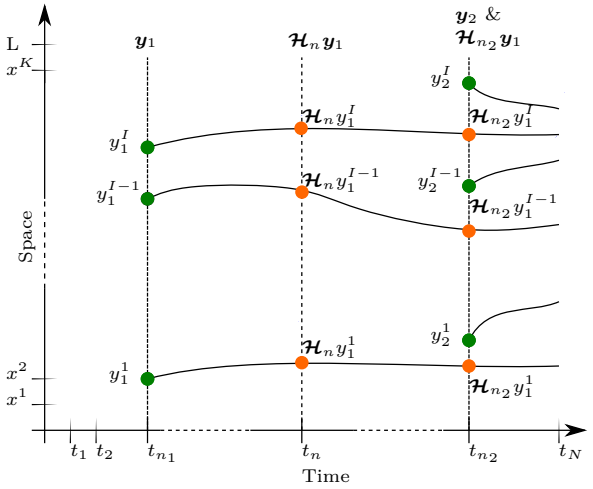


Figure 4.1: Schematic depiction of how locations of pseudo-observations $\mathcal{H}_n\boldsymbol{y}_m$(orange) are derived from the locations of observations $\boldsymbol{y}_m$(green) by propagating along characteristics(black). In this graphic at t_{n_1}, only observations are available, thus only a classical filtering step can be performed, at t_n only pseudo-observations are available, thus a regular DLF step can be performed, and at t_{n_M} observations and pseudo-observations are available, which means an MDLF step would be performed. (See section 4.5.1 for the definition of DLF/MDLF-step and calculations of $\mathcal{H}_n\boldsymbol{y}_m$ as well as $\mathcal{H}_n\boldsymbol{Y}_m$.) (In text mentions: p.107)

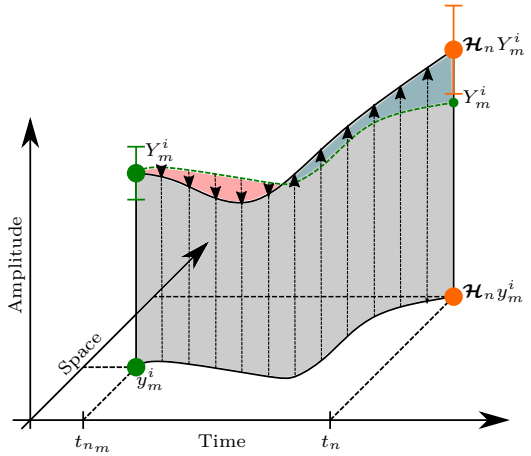


Figure 4.2: Schematic depiction of how values of pseudo-observations $\mathcal{H}_n\boldsymbol{Y}_m$ are derived by evolving observations $\boldsymbol{Y}_m$ along characteristics. At t_{n_m}, we see a single observation Y_m^i at location y_m^i(green dots). At t_n the pseudo-observation $\mathcal{H}_n Y_m^i$ at location $\mathcal{H}_n y_m^i$ is depicted(orange dots.) As a dashed green line, we see a curve of constant amplitude Y_m^i along the characteristic starting at y_m^i. From Y_m^i to $\mathcal{H}_n Y_m^i$ in black the actual trajectory of the pseudo-observation is shown. Underlayed in red are times, when the pseudo-observation is smaller than Y_m^i, and underlayed in blue are times, at which the pseudo-observation exceeds the value of Y_m^i. (In text mentions: p.107)

$$\tilde{u}^i(t_{n_m}) = Y^i_m, \tag{4.19}$$

Provided $v_{(x)}$ and $v_{(xx)}$ along the characteristic $\boldsymbol{x}_m(t)$, equation (4.19) is a well-posed approximation of (4.17). We thus finally define the pseudo-observations

$$\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m := \left(x^i_m(t_n)\right)^I_{i=1} \tag{4.20}$$

and

$$\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m := \left(\tilde{u}^i(t_n)\right)^I_{i=1}, \tag{4.21}$$

where the $x^i_m(t)$ and $\tilde{u}^i(t)$ solve equations (4.17) and (4.19) respectively for $i = 1, \ldots, I$. At estimation times $t_n \in T$, the pseudo-observations $(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m)$ are treated like real observations, and the *pseudo-observation error* $\boldsymbol{\zeta}_n \in \mathbb{R}^I$ is defined as

$$\boldsymbol{\zeta}_n = \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m - v(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m, t_n). \tag{4.22}$$

$\boldsymbol{\zeta}_n$ is assumed to be distributed according to a mean zero Gaussian distribution. Recall that the value of $v(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m, t_n)$ can be obtained from $\boldsymbol{V}_n$ using (4.6). Equation (4.22) thus induces the distribution $\pi(\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m | \boldsymbol{V}_n)$, which will be used to inform the Likelihood at times t_n used in the DLF and discussed in section 4.5.2. For later reference, we define $\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_{\ell:m}$ and $\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{\ell:m}$ for $n_\ell < n_m < n$ as, respectively, the column vectors obtained by concatenating all $\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_{n_{m'}}$ and $\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{n_{m'}}$ with $t_{n_{m'}} \in T_O$ and $t_{n_\ell} \leq t_{n_{m'}} \leq t_{n_m}$. Namely,

$$\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_{\ell:m} = \begin{pmatrix} \boldsymbol{\mathcal{H}}_n \boldsymbol{y}_\ell \\ \vdots \\ \boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m \end{pmatrix}, \qquad \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{\ell:m} = \begin{pmatrix} \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_\ell \\ \vdots \\ \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m \end{pmatrix}$$

Thus $(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_{\ell:m}, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{\ell:m})$ contains all pseudo-observations at time t_n derived from observations that were measured at times between t_{n_ℓ} and t_{n_m}.

4.5.2 Formulation of the Filter

We define for the duration of this section $n \in \{1, \dots, N\}$ and $m \in \{1, \dots, M\}$ such that $n_m \leq n \leq n_{m+1}$. Under the assumption posed on the model, observations and pseudo-observations, the DLF yields an alternative form of the posterior of the model, up to time t_n, conditioned on observations measured up to time t_{n_m}:

$$\pi(\boldsymbol{V}_{0:n}|\boldsymbol{Y}_{1:m}) \propto \pi(\boldsymbol{Y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m-1}|\boldsymbol{V}_n)\pi(\boldsymbol{V}_{0:n}|\boldsymbol{Y}_{1:m-1}). \tag{4.23}$$

Let $\boldsymbol{V}_{n|n}$ denote the posterior of $\boldsymbol{V}_n$ conditioned on observations up to time t_n. It is distributed as

$$\begin{aligned}
\boldsymbol{V}_{n|n} &\sim \pi(\boldsymbol{V}_n|\boldsymbol{Y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m-1}) \\
&= \int \pi(\boldsymbol{V}_{0:n_m}|\boldsymbol{Y}_{1:m}) d\boldsymbol{V}_{n_m-1} \cdots d\boldsymbol{V}_0 \\
&\propto \pi(\boldsymbol{Y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m-1}|\boldsymbol{V}_{n_m}) \cdot \int \pi(\boldsymbol{V}_{0:n_m-1}|\boldsymbol{Y}_{1:m-1}) d\boldsymbol{V}_{n_m-1} \cdots d\boldsymbol{V}_0 \\
&= \pi(\boldsymbol{Y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m-1}|\boldsymbol{V}_n)\pi(\boldsymbol{V}_n|\boldsymbol{V}_{n-1|n-1}).
\end{aligned} \tag{4.24}$$

We therefore see that this posterior at time t_n is entirely determined by the likelihood of observations and pseudo-observations at t_n conditioned on the model up to this point $\pi(\boldsymbol{Y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m-1}|\boldsymbol{V}_n)$, and the distribution of the model conditioned on the posterior at the previous time step $\pi(\boldsymbol{V}_n|\boldsymbol{V}_{n-1|n-1})$. The latter is used as a prior at t_n.

This shows that, as in the KF approach and given the observations and pseudo-observations, the distribution of $\boldsymbol{V}_{n|n}$ can be found sequentially based on the posterior distribution at the previous time step $\boldsymbol{V}_{n-1|n-1}$. For $t_n \leq \min(T_O)$ there are no pseudo-observations, i.e. $\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:n-1} = (\ )$ is an empty vector, thus the KF approach is recovered.

If on the other hand $t_n > \min(T_O)$ there are two cases to be considered:

(i) there are no observations at $t_n \notin T_O$ and thus $\pi(\boldsymbol{Y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m-1}|\boldsymbol{V}_n) = \pi(\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m-1}|\boldsymbol{V}_n)$,

(ii) $t_n = t_{n_m} \in T_O$ and therefore
$\pi(\boldsymbol{Y}_m, \boldsymbol{\mathcal{H}}_{n_m} \boldsymbol{Y}_{1:m-1}|\boldsymbol{V}_{n_m}) = \pi(\boldsymbol{\mathcal{H}}_{n_m} \boldsymbol{Y}_{1:m-1}|\boldsymbol{V}_{n_m})\pi(\boldsymbol{Y}_m|\boldsymbol{V}_{n_m})$.

We call a step of the DLF algorithm in the first case a *DLF update* and in the second case a *multi analysis DLF (MDLF) update*. In the first case one proceeds as follows:

The DLF update:

$$\begin{aligned}
\text{Predict:}\langle \boldsymbol{V}_{n|n-1}\rangle &= \boldsymbol{L}_{n-1}\langle \boldsymbol{V}_{n-1|n-1}\rangle + \Delta t f_{n-1} \\
\boldsymbol{P}_{n|n-1} &= \boldsymbol{L}_{n-1}\boldsymbol{P}_{n-1|n-1}\boldsymbol{L}_{n-1}^{\top} + \boldsymbol{Q}_{n-1} \\
(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}}, \boldsymbol{Y}_{\boldsymbol{\mathcal{H}}}) &= (\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_{1:m}, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m}) \\
\boldsymbol{R}_{\boldsymbol{\mathcal{H}}} &= \mathrm{Cov}(\boldsymbol{Y}_{\boldsymbol{\mathcal{H}}}, \boldsymbol{Y}_{\boldsymbol{\mathcal{H}}}) \\
\text{Analysis:}\langle \boldsymbol{V}_{n|n}\rangle &= \boldsymbol{V}_{n|n-1} + \boldsymbol{K}_n \left(\langle \boldsymbol{Y}_{\boldsymbol{\mathcal{H}}}\rangle - \boldsymbol{H}(\boldsymbol{y})\langle \boldsymbol{V}_{n|n-1}\rangle\right) \\
\boldsymbol{P}_{n|n} &= (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}}))\boldsymbol{P}_{n|n-1}
\end{aligned}$$

where
$\boldsymbol{K}_n = \boldsymbol{P}_{n|n-1}\boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})^{\top}\left(\boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})\boldsymbol{P}_{n|n-1}\boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})^{\top} + \boldsymbol{R}\right)^{-1}$ This covers the case, where only pseudo-observations are available at time t_n. Derivations of this mirror the derivation of the KF exactly, if observations are replaced by pseudo-observations. If both real and pseudo-observations are available at $t_n = t_{n_m}$, i.e. $n = n_m$, a multi-analysis step is performed. The prediction step is identical as in the previous case. The Analysis step however changes as follows:

The MDLF update:

$$\begin{aligned}
\text{Predict:}\langle \boldsymbol{V}_{n_m|n_m-1}\rangle &= \boldsymbol{L}_{n_m-1}\langle \boldsymbol{V}_{n_m-1|n_m-1}\rangle + \Delta t f_{n_m-1} \\
\boldsymbol{P}_{n_m|m_m-1} &= \boldsymbol{L}_{n_m-1}\boldsymbol{P}_{n_m-1|n_m-1}\boldsymbol{L}_{n_m-1}^{\top} + \boldsymbol{Q}_{n_m-1} \\
(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}}, \boldsymbol{Y}_{\boldsymbol{\mathcal{H}}}) &= (\boldsymbol{\mathcal{H}}_{n_m} \boldsymbol{y}_{1:m-1}, \boldsymbol{\mathcal{H}}_{n_m} \boldsymbol{Y}_{1:m-1}) \\
\boldsymbol{R}_{\boldsymbol{\mathcal{H}}} &= \mathrm{Cov}(\boldsymbol{Y}_{\boldsymbol{\mathcal{H}}}, \boldsymbol{Y}_{\boldsymbol{\mathcal{H}}}) \\
\text{Update:}\langle \boldsymbol{V}_{n_m|n_m}\rangle &= \langle \boldsymbol{V}_{n_m|n_m-1}\rangle + \boldsymbol{K}^*_{n_m}(\langle \boldsymbol{Y}_m\rangle - \boldsymbol{H}(\boldsymbol{y}_m)\langle \boldsymbol{V}_{n|n-1}\rangle) \\
&\quad + \boldsymbol{J}_{n_m}\left(\langle \boldsymbol{Y}_{\boldsymbol{\mathcal{H}}}\rangle - \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})\langle \boldsymbol{V}_{n|n-1}\rangle\right) \\
\boldsymbol{P}_{n_m|n_m} &= (\boldsymbol{I} - \boldsymbol{K}^*_{n_m}\boldsymbol{H}(\boldsymbol{y}_m)
\end{aligned}$$

$$- \boldsymbol{J}_{n_m} \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})) \boldsymbol{P}_{n_m|n_m-1}$$

where

$$\boldsymbol{K}^*_{n_m} = (\boldsymbol{I} - (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}(\boldsymbol{y}_m)) D \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})) \boldsymbol{K}$$
$$\boldsymbol{J}_{n_m} = (\boldsymbol{I} - \boldsymbol{K}_{n_m} \boldsymbol{H}(\boldsymbol{y}_m)) \boldsymbol{D}$$

with

$$\begin{aligned}
\boldsymbol{K} =& \boldsymbol{P}_{n_m|n_m-1} \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})^\top \cdot (\boldsymbol{R}_{n_m} + \boldsymbol{H}(\boldsymbol{y}_m) \boldsymbol{P}_{n_m|n_m-1} \boldsymbol{H}(\boldsymbol{y}_m)^\top)^{-1} \\
\boldsymbol{D} =& \boldsymbol{P}_{n_m|n_m-1} \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})^\top \Big(\boldsymbol{R}_{\boldsymbol{\mathcal{H}}} + \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}}) \boldsymbol{P}_{n_m|n_m-1} \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})^\top \\
& - \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}}) \boldsymbol{K}_{n_m} \boldsymbol{H}(\boldsymbol{y}_m) \boldsymbol{P}_{n_m|n_m-1} \boldsymbol{H}(\boldsymbol{y}_{\boldsymbol{\mathcal{H}}})^\top \Big)^{-1}.
\end{aligned}$$

The derivation of these gains can be found in either [13] or [5]. Note that the size of the matrices to invert in the calculation of these gains is determined by the size of $\boldsymbol{R}_{\boldsymbol{\mathcal{H}}} \in \mathbb{R}^{mI \times mI}$. In practice, it might be reasonable to limit the size of this matrix by not using all pseudo-observations indefinitely, but rather discarding some in a trade-off between accuracy and complexity. As an example, replacing $\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_{1:m}$ and $\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m}$ by $\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_{k_n:m}$ and $\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{k_n:m}$ respectively in the previous algorithms, for some k_n, would discard pseudo-observations derived from the oldest observations as time goes on, thus limiting the size of $\boldsymbol{R}_{\boldsymbol{\mathcal{H}}}$ to $(m - k_n)I(m - k_n)I$. The overall complexity of the DLF and the effects of discarding "older" pseudo-observations like this are discussed in the next section.

4.6 Analysis of Computational Complexity of the DLF

The DLF approach has an added computational overhead, as compared to its standard counterpart. In what follows we estimate the overhead of the DLF approach to Kalman filtering, as compared to the native Kalman filter. As will be shown subsequently, improvements in the estimates obtained using DLF may offset the added computational

burden. The additional computational cost of the DLF, compared to the model on its own, stems from two sources:

1. Calculating the pseudo-observations $(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_{1:m-1}, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_{1:m-1})$.

2. Calculating and applying the gains $\boldsymbol{K}_n, \boldsymbol{K}_n^*, \boldsymbol{J}_n$.

If an explicit ODE solver is used to solve equations (4.17) and (4.19) to calculate $(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m)$ at $t_{n_m} \in T_O$ the complexity is linear in the number of time steps $\frac{t_n - t_{n_m}}{\Delta t}$. Since $\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m \in \mathbb{R}^I$ and $\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m$ is normally distributed in $\mathbb{R}^I$ these calculations over a single time step are dominated by the calculation of the covariance $\boldsymbol{R}_{n,m} = \mathrm{Cov}(\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m) \in \mathbb{R}^{I \times I}$ leading to an overall complexity of order $O\left(\frac{t_n - t_{n_m}}{\Delta t} I^2\right)$ to calculate $(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m)$ for a single $t_{n_m} \in T_O$. Considering this has to be done for all pseudo-observations $(\boldsymbol{\mathcal{H}}_N \boldsymbol{y}_{1:M-1}, \boldsymbol{\mathcal{H}}_N \boldsymbol{Y}_{1:M-1})$ this leads to a complexity of $O\left(\sum_{m=1}^{M} \frac{t_N - t_{n_m}}{\Delta t} I^2\right) \leq O\left(MNI^2\right)$.

The complexity of calculating the gains at time $t_n \in T$ is dominated by inverting matrices of the same size as the covariance matrix $\boldsymbol{R}_{\boldsymbol{\mathcal{H}}}$. At time t_n let this size be $s_n \times s_n$. Note that for times $t_{n_m} \leq t_n < t_{n_{m+1}}$ this size is $s_n = mI$. The computational cost of matrix inversion is cubic in the number of rows and therefore the complexity for the $n-$th time step is $O(I^3 M^3)$. Over all time steps this is bounded by $O(NM^3 I^3)$. The complexity of the KF arises from inverting matrices of size $I \times I$ at each of the M observation times, thus having an overall complexity of $O(MI^3)$. The DLF's overall complexity is $O(MI^3 + MNI^2)$.

Note however, that in all these complexity estimates so far we assumed that all pseudo-observations $(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m)$ are used for all $t_n > t_{n_m}$. As $t_n - t_{n_m}$ increases so does the uncertainty associated with $(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m)$. Therefore discarding this pseudo-observation eventually would have only a small effect on overall accuracy. Thus if run time is of the essence, discarding some pseudo-observations after they outlived their usefulness can help to keep complexity in check. One way to proceed with this program is to set an upper bound for the number of pseudo-observations assimilated at any given time, or by setting a threshold on uncertainty, discarding the relatively most uncertain observations/pseudo-observations.

As an example, let us assume the number of pseudo-observations is capped at an integer multiple of I, say pI, and the oldest pseudo-observations are discarded every time this threshold is reached. This means at any given time t_n only observations from the last p

observation times are used to calculate pseudo-observations. At an observation $t_n = t_{n_m}$, a multi-analysis step is performed, before the oldest pseudo-observations $(\boldsymbol{\mathcal{H}}_{n_m}\boldsymbol{y}_{m-p}, \boldsymbol{\mathcal{H}}_{n_m}\boldsymbol{Y}_{m-p})$ from time $t_{n_{m-p}}$ are discarded. In other words $(\boldsymbol{\mathcal{H}}_n\boldsymbol{y}_{1:m-1}, \boldsymbol{\mathcal{H}}_n\boldsymbol{Y}_{1:m-1})$ in the DLF algorithm is replaced by $(\boldsymbol{\mathcal{H}}_n\boldsymbol{y}_{m-p:m-1}, \boldsymbol{\mathcal{H}}_n\boldsymbol{Y}_{m-p:m-1})$. This limits the number of pseudo-observations concurrently in the algorithm to pI from the previous maximum of MI. We get new complexity estimates under these circumstances by making the replacement $M \to p$ in the previous estimates. Thus the complexity of the DLF falls to $O(Np^3I^3 + pNI^2)$. Assuming p is picked reasonably small ($p \ll N$ and $p \ll I$) this reduces further to $O(NI^3)$, which is comparable to a standard KF's complexity of be $O(MI^3)$. (While the calculation of the gains can be a bottleneck of the algorithm, it is noteworthy that as long as neither c nor f in equations (4.2) and (4.3) depend on the observations $\boldsymbol{Y}_{1:M}$, the gains are also independent of $\boldsymbol{Y}_{1:M}$ and can thus be calculated offline).

4.7 Numerical Results and Comparisons

We contrast the DLF approach, as applied to the KF (the DLF), with estimates obtained by the native KF (the KF). We will also discuss the numerical details of our implementation and introduce the metrics, based on which we will evaluate the performance of the two approaches.

4.7.1 Computational Details

Since the DLF approach was developed specifically for hyperbolic (wave) equations, we are especially interested in determining to what extent it can handle advection as well as macroscale diffusion. We thus introduce the non-dimensional quantity $\alpha := D/c_0L$, where D is the diffusion coefficient, c_0 is the typical size of the velocity C, and L is the characteristic length which is taken as the length of the domain, to capture the extent to which diffusion processes and advection qualitatively affect the solution. Let the primed nondimensional quantities be

$$x' = \frac{x}{L}, \quad t' = \frac{tc_0}{L}, \quad u' = \frac{u}{u_0}, \quad F' = \frac{FL}{u_0c_0}, \quad C' = \frac{C}{c_0},$$

then Eq. (4.1) is now recast as

$$\begin{aligned} u_t - (c+\chi)u_x &= \alpha u_{xx} + f + \phi, \quad t > 0, x \in [0,1], \\ u(x,0) &= \mathcal{U}(x), \quad x \in [0,1], \end{aligned} \tag{4.25}$$

in dimensionless units, having dropped the primes. It is still assumed that ϕ and χ are noise terms generated by Wiener processes. Therefore $\chi dt = AdW^c$ and $\phi dt = BdW^u$ still hold.

We run all numerical examples shown in this section on a domain $[0,1]$ with periodic boundary conditions and times spanning $t_0 = 0$ to $t_N = 0.5$. We chose $\Delta x = 0.01$ and $\Delta t = 0.005$, in dimensionless units for the discretization. The wave velocity is set to be

$$C(x,t)dt = \cos(5\pi t)\, dt + AdW^c(x,t) + \tilde{A}d\tilde{W}_c(t)$$

Note that we split the wave noise term into $AdW^c(x,t) + \tilde{A}d\tilde{W}_c(t)$. Both dW^c and $d\tilde{W}_c$ are incremental Wiener processes, but dW^c will be assumed to be uncorrelated in space, *i.e.*, $\langle dW^c(x)dW^c(y)\rangle = \delta_{x,y}$ for all $x, y \in X$, while $d\tilde{W}_c$ is independent of x. We will set $A = 0.05$ for all numerical experiments but will consider two cases of $\tilde{A}$ when simulating the truth. In the first case $\tilde{A} = 0$, while in the second $\tilde{A} = 1$. The model will be unaware of $\tilde{A}$, i.e. assume $\tilde{A} = 0$ in both cases. This introduces a systematic error in C for the model that the KF and DLF will have to overcome. The effects of this will be discussed in section 4.7.2.

In all of the following examples, we assume that the forcing is given by

$$F(x,t)dt = BdW^u(x,t) = 0.05dW^u(x,t)$$

where $dW^u(x,t)$ is an incremental Wiener process in time for each x and $\langle dW^u(x,t)dW^u(y,t)\rangle = \delta_{x,y}$ for all $x, y \in X$.

The initial data will be

$$u_0(x) = \sigma \exp(-250(x-\theta)^2).$$

The amplitude σ and phase θ will be deterministic or chosen from random distributions. We will discuss three kinds of initial data for the model: (i) the deterministic case where $\sigma = 1$ and $\theta = \frac{1}{2}$ (see Section 4.7.2); (ii) with uncertain amplitude: $\sigma \sim \mathcal{U}\left[\frac{1}{2}, \frac{3}{2}\right]$, where $\mathcal{U}$ indicates a uniform distribution, and $\theta = \frac{1}{2}$; and (iii) the case of $\sigma = 1$ and uncertain phase $\theta \sim \mathcal{U}[0, 1]$. In all these cases the amplitude and phase of the truth are fixed to $\sigma = 1$ and $\theta = \frac{1}{2}$. Cases (ii) and (iii) will be highlighted in Section 4.7.2.

The *truth* will be used to test the outcomes as well as to generate observations. The truth is computed through Strang-splitting [19]. Equation (4.1) is split into a noisy advection equation

$$u_t - (c + \chi)u_x = f + \phi$$

and a deterministic diffusion equation

$$u_t = \alpha u_{xx},$$

which are then used to generate a solution sequentially. The diffusive step is solved via FFT by calculating an exact solution in Fourier space. For the advective part of the splitting, we chose a Lax-Wendroff scheme in space and a stochastic Runge-Kutta scheme in time. The chosen RK method is a third-order scheme with second-order weak convergence [16].

The *model* is given to us as a first-order split step procedure, splitting (4.1) into the same noisy advection equation and diffusion equation as for the truth. The diffusion step is, again, solved exactly via FFT. The advective step of the model is solved with an upwind scheme in space and an explicit Euler scheme in time. The Courant number for all methods is 1.

Observations are derived from the truth. These are made available at observation times $t_{n_m} \in \{0.05, 0.1, 0.15, ..., 0.45\} = T_O$ by uniformly drawing their locations $y_m^1, \ldots, y_m^I \in X$ from the grid, without repetition. Their corresponding values are then determined from the truth via $Y_m^i = u(y_m^i, t_{n_m}) + \epsilon_m^i$, where the ϵ_m^i are drawn independently from a mean zero normal distribution with variance 10^{-4}. Throughout the next sections, the number of observations at each observation time will take values $I \in \{10, 20, 40, 60\}$.

Both the KF and the DLF require the interpolation operator $\boldsymbol{H}(x)$. In these tests, we use a simple linear interpolation operator, which for $x \in [0, L]$ is defined as

$$\boldsymbol{H}(x) := ((1 - r(x)\delta_{k,\ell}) + r(x)\delta_{k,\ell+1})_{k=0}^{K} \in \mathbb{R}^{1\times K}. \tag{4.26}$$

Here, $r(x) = \mod(x, \Delta x)$ is the remainder of $\frac{x}{\Delta x}$, $\delta_{i,j}$ is the Kronecker delta, and $\ell \in \{1, \ldots, K\}$ is chosen such that $x^\ell \leq x < x^{\ell+1}$ for grid points $x^\ell, x^{\ell+1} \in X$. (To account for periodic boundary conditions, assume $x^K < x^0 = L$ for $x^K \leq x < L$).

To calculate $\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m$, we use an explicit Euler algorithm to solve equation (4.17). Since for $\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m$, the calculation of the mean and covariance is sufficient for the algorithm we use an explicit Euler algorithm based on equation (4.19) for each of those as well. The derivatives of the model are therefore only required at $t_n \in T$. We obtain them through the model as $v_{(x)}(x, t_n) = \boldsymbol{H}(x)\nabla_x \boldsymbol{V}_n$ and $v_{(xx)}(x, t_n) = \boldsymbol{H}(x)\nabla_{xx} \boldsymbol{V}_n$, where ∇_x and ∇_{xx} are center difference operators approximating first and second derivatives. Thus over a single time step from $t_{n-1} \geq t_m$ to t_n and with $\tilde{\boldsymbol{H}}_n := \boldsymbol{H}(\boldsymbol{\mathcal{H}}_n \boldsymbol{y}_m)$ and $\boldsymbol{R}_{n,m} = \mathrm{Cov}(\boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m, \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m)$ we use

$$\langle \boldsymbol{\mathcal{H}}_n \boldsymbol{Y}_m \rangle = \langle \boldsymbol{\mathcal{H}}_{n-1} \boldsymbol{Y}_m \rangle + \Delta t \cdot \left(D + \frac{A^2}{2} \right) \tilde{\boldsymbol{H}}_{n-1} \nabla_{xx} \langle \boldsymbol{V}_{n-1|n-1} \rangle,$$

and

$$\begin{aligned} \boldsymbol{R}_{n,m} = \boldsymbol{R}_{n-1,m} &+ \Delta t \cdot \left(B^2 \boldsymbol{I} + A^2 (\nabla_x \tilde{\boldsymbol{H}}_{n-1} \langle V_{n-1|n-1} \rangle)(\nabla_1 \tilde{\boldsymbol{H}}_{n-1} V_{n-1|n-1})^\top \right) \\ &+ \Delta t^2 \cdot \left(D + \frac{A^2}{2} \right) \tilde{\boldsymbol{H}}_{n-1} \nabla_{xx} \boldsymbol{P}_{n-1|n-1} \nabla_{xx}^\top \tilde{\boldsymbol{H}}_{n-1}^\top. \end{aligned} \tag{4.27}$$

The second line in (4.17) is an explicit Euler scheme modeling the system noise, while the third line tracks the errors introduced due to the uncertainty of $\boldsymbol{V}_n$.

To quantitatively compare the traditional KF to the DLF approach to the KF we calculate the following metrics: the Residual Mean Square (RMS) error, the Mass error, the Center of Mass (CoM) error, and the probabilistic Calibration. These are given by

$$\text{RMS error:} \sqrt{\Delta t \Delta x \sum_{n=1}^{N} \sum_{k=1}^{K} |U^i(t_n) - \langle V_{n|n}^k \rangle)|^2}$$

$$\text{Mass error:}\sqrt{\Delta t \Delta x^2 \sum_{n=1}^{N} \left| \sum_{i=1}^{N} |U^i(t_n)| - \sum_{i=1}^{N} |\langle V_{n|n}^k \rangle| \right|^2}$$

$$\text{CoM error:}\sqrt{\Delta t \sum_{n=1}^{N} \left| \frac{\sum_{k=1}^{K} |U^k(t_n) x^k|}{\sum_{k=1}^{K} |U^k(t_n)|} - \frac{\sum_{k=1}^{K} |\langle V_{n|n}^k \rangle x^k|}{\sum_{k=1}^{K} |\langle V_{n|n}^k \rangle|} \right|^2}$$

$$\text{Calibration:}\frac{\Delta t}{t_N N} \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{1}\left((U^k(t_n) - \langle V_{n|n}^k \rangle) < 2\sqrt{\text{Var}(V_{n|n}^k)} \right),$$

where $\mathbb{1}(\cdot)$ is the indicator function.

The RMS error tracks the sum of local errors between model and truth, while the Mass error determines how accurately the total mass in the system is captured. The CoM error remains small if the position of the center of mass is captured well by the model. This is important in advection-dominated problems and in our examples will be mostly determined by how well the position of the maximum is captured over time. The Calibration measures the percentage of times the truth is within two of the estimated standard deviations of the model. If both noise and model error are normally distributed and captured correctly by the uncertainty, this value should be approximately 95%. Higher values indicate the variance is overestimated, while smaller values mean the uncertainty is larger than estimated. In the following sections, we will see these measures evaluated in total and at specific times. When evaluated at a specific time t_n, the Δt and the sum over n will be dropped and what remains will be evaluated at the corresponding n. To account for the randomness in the generation of the truth,(and the initial data of the model, where applicable,) all records of these four metrics from hereon will be based on 50 runs each: Line plots of RMS, Mass, Com error and Calibration will show their mean value over 50 runs, while all box plots will be based on their respective minimum, maximum and the $25\%, 50\%$ and 75% quantiles. Between each of these runs the truth, observations, and initial data will be regenerated. For comparability, the KF and DLF will use the same Observations and initial data for each individual run.

4.7.2 Comparing the KF, and the DLF Outcomes

We will compare outcomes for deterministic and aleatoric initial data, uncertainties in the phase speed and as a function of α.

Comparing the KF and DLF in Systems with Deterministic Initial Data

We will compare posterior predictions $\boldsymbol{V}_{n|n}$. The comparison is conducted in a setting where both the noise of the phase speed dW^c and the noise of the forcing dW^u are uncorrelated in space, i.e., $\langle dW(x)dW(y)\rangle = \delta_{x,y}$. Both methods are provided with deterministic initial data, $\sigma = 1$, and $\theta = \frac{1}{2}$ and $\tilde{A} = 0$. Throughout this comparison, both the relative diffusion α and the number of data points per observation I will be varied. We will demonstrate that the DLF outperforms the KF in terms of RMS, Mass error, and Calibration, particularly when the number of data points is sparse and when α is small. The amplitude of the wave noise will be fixed at $A = 0.05$ for the entirety of this section.

We first examine an individual run in the advection-dominated case. For $\alpha = 0.01$, Figure 4.3 shows the truth (right), the model prediction through the KF (left), and the prediction of the DLF (middle). Both filters were presented with $I = 20$ data points per observation time. The location of these data points is randomly selected at each observation time but is identical between the KF and the DLF. Note that the trajectories of pseudo-observations depicted extend beyond the availability of observation, providing Bayesian predictions at times $t > 0.45$, which can be considered the future.

At the observation sites, both the KF and the DLF pick up the values of the observed data and adjust their predictions during the analysis step. The DLF manages to maintain these adjusted values over the simulation time, while in the KF, adjustments due to observations quickly vanish due to diffusion. This is not surprising, as the DLF reinforces the information gathered from observations through pseudo-observations that move along characteristics.

Figure 4.4 confirms that the DLF captures the truth more accurately, as quantified by the metrics. We examine the time series of these four metrics for the same parameters and conditions used to generate the previous example. Starting from time $t = 0.05$, the first observation time, there is a clear divide between the KF and the DLF in the RMS and

Mass errors, with the DLF performing significantly better. The same can be said for the Calibration, though the advantage of the DLF is less pronounced. There appears to be no such clear trend for the CoM error.

As the last part of this set of experiments, we examine if these trends hold up when the remaining parameters α and I are varied. Figure 4.5 depicts boxplots of the time-averages of the four metrics across a range of numbers of observations I and diffusion constants α. All combinations of $I \in \{10, 20, 40, 60\}$ and $\alpha \in \{0.001, 0.01, 0.1\}$ were analyzed. These numerical examples confirm our expectations: for advection-dominated dynamics and sparse but low-uncertainty observations, the DLF does significantly better in terms of Mass errors. In terms of the RMS, CoM, and Calibration, we see that the DLF outperforms the KF when data is sparse, namely $I = 10, 20$. The KF gains an advantage when observations are plentiful.

Comparison of the KF and DLF Estimates When Uncertainties in the Initial Conditions Are Present

Since the DLF constantly imparts phase information via the likelihood of the pseudo-observations conditioned on the model, the DLF approach should deliver better predictions than the KF on problems where there are uncertainties in the initial data. Again, this is expected when the data has low uncertainty and the dynamics are advection dominated. We will compare the quality metrics of the two methods with initial condition uncertainty. We will show that the DLF manages to overcome this restriction within a few time steps, practically reaching error values comparable to the case of known initial data.

To simulate this uncertainty in the initial data, the truth will be generated with the same initial data as in the previous section, namely $\sigma = 1$ and $\theta = 0.5$, while both the KF and the DLF will be provided different initial data. We will examine two different cases in this section. First, the amplitude σ provided to the filters will be drawn uniformly from $\mathcal{U}[\frac{1}{2}, \frac{3}{2}]$, unless individual runs are discussed, for which $\sigma \neq 1$ is picked by hand. Throughout this first set of examples, the phase $\theta = \frac{1}{2}$ is assumed known. In the next set of examples, the phase θ will be uniformly drawn from $\mathcal{U}[0, 1]$, while assuming $\sigma = 1$ is known. There will again be an

exception when discussing individual runs, for which $\theta \neq 0.5$ is picked by hand. The initial data provided to the KF and DLF will be identical for each run to guarantee comparability.

We test the same values for the relative diffusion $\alpha = 0.01$ and the number of data points per observation $I = 20$ as in the previous section. Noise levels remain the same as in the previous section. We will demonstrate that the DLF is significantly more successful in correcting its incomplete knowledge of the initial data than the KF. All trends observed in the previous section, that is, smaller errors and better calibration for the DLF, persist under these settings.

In Figure 4.6, we see the results of an individual run where σ was set to 0.7 for the model and $\theta = \frac{1}{2}$. The number of observations available is $I = 20$, and $\alpha = 0.01$.

We see that the DLF manages to correct its incorrect initial amplitude almost immediately as soon as it has access to observations, while the KF struggles to correct its estimation of the amplitude throughout the run.

The time series of the four metrics are shown in Figure 4.7. These still used $I = 20$ nor $\alpha = 0.01$ was changed from the previous run. We observe qualitatively similar results as before. The DLF performs better in terms of RMS and Mass error, as well as in Calibration. There is no significant difference in the CoM error.

The advantage of the DLF over the KF in terms of RMS and Mass error is significant. After a brief adjustment period, the DLF reaches the same levels as in the case with known initial data. The advantage in Calibration is less pronounced and comparable to the previous case.

Figure 4.8 shows boxplots of the total value of each of the four metrics across parameters $\alpha \in \{0.001, 0.01, 0.1\}$ and $I \in \{10, 20, 40, 60\}$. We observe the DLF performing better on RMS and Mass error, as well as Calibration, with the advantage expectedly dwindling as the number of observations increases and the diffusion ramps up. In the case of the Mass error, the DLF is still superior across all I and α.

Next, we will focus on phase error effects on the estimation. In the remainder of this section, we repeat the previous experiments, but now fix $\sigma = 1$, while $\theta \sim \mathcal{U}[0, 1]$, unless focusing on just a single run.

Figure 4.9 depicts results from such an individual run with $\theta = 0.25$. The diffusion was, again, set to $\alpha = 0.01$ and $I = 20$. Both filters start with the mode in the wrong position and pick up on their location error as observations become available. The initially displaced mode decreases in amplitude for both models, while a second mode in the correct location starts emerging as soon as observations are available.

The DLF manages to suppress the wrong mode over just a few time steps, roughly the same amount of time it takes to pick up the correct position and amplitude of the actual mode. The KF does significantly worse in this setting, taking almost the entire simulation time to drop the phase error.

In Figure 4.10, we analyze the average time series of four quality metrics again. The diffusion $\alpha = 0.01$ and number of observations $I = 20$ remain the same. The DLF still performs better than the KF in terms of RMS error and Calibration, with a much bigger advantage in Calibration than in the previously discussed examples. For the first time, there is a clear difference in the CoM error, with the DLF taking a significant lead. Note that in terms of the Mass error, the DLF initially does worse than the KF. This can be explained by the DLF picking up the phase and amplitude of the correct mode, before phasing out the incorrect first mode. In fact there is a brief period where it estimates the existence of two modes. In the long run, the DLF outperforms in terms of Mass error as well.

To close out this section, we test whether the advantage of the DLF over the KF can be sustained for different values of I and α. Figure 4.11 depicts the statistics of the four metrics for $I \in \{10, 20, 40, 60\}$ and $\alpha \in \{0.001, 0.01, 0.1\}$. We observe the following: The DLF does substantially better on RMS, CoM, and Calibration, confirming all trends seen so far in this section. Unlike previously observed, the DLF sustains its advantage throughout cases with higher numbers of observations, likely because more observations make it more probable to pick up the correct location of the true mode in the analysis stage of the assimilation.

In terms of Mass, the DLF now actually performs worse than in previous cases, and even slightly worse than the KF, when few observations are available. This can be explained by the fact that we are looking at time averages of the Mass error here. Since the DLF yields two mode estimates early on, for a brief period, its Mass error is higher. Additional observations help depress the second mode.

Comparing the KF and DLF When Uncertainties in Phase Speed are Present

In the previous section, we compared the DLF to the KF assuming limited knowledge of the initial data. In this section, we will showcase how both filters perform under the assumption that phase errors are significant. To this end, during the simulation of the truth, $\tilde{A} = 1$ will be used, while the model and thus KF and DLF estimators are unaware of this and still assume $\tilde{A} = 0$. This will cause substantial divergence between the phase speed of the truth and the phase speed used by the model, resulting in significant displacement of the position of the center of mass, if no assimilation happens. Initial data is assumed to be known. The DLF outperforms the KF in terms of RMS and Mass metrics, but it will also be shown that the DLF can correct the displaced center of mass better than the KF. We first take a look at an individual run again. Predictions and truth are depicted in Figure 4.12. Diffusion is again $\alpha = 0.01$ and $I = 20$ observations are available at each observation time for this example. As seen in previous sections the DLF manages to adjust its predictions to the observations much more rapidly than the KF.

Regarding the time series of the quality metrics depicted in Figure 4.13 we now see the DLF outperforming the traditional KF in all metrics except the CoM error, where it occasionally does slightly worse.

Considering the total values of the four metrics over a larger range of α and I we see the DLF clearly taking the lead. Boxplots of these statistics are depicted in figure 4.14. The DLF performs better on average regarding all four metrics, this time persisting through higher diffusion and increased number of observations, as far as examined. This again shows the superiority of the DLF over the KF in the case of an ill-informed model, this time illustrated by the models getting the phase speed wrong.

Dynamics and the DLF and KF outcomes

In previous sections, we compared the capabilities of the DLF and the KF under increasingly complex uncertainties in the model. All these experiments were initially conducted in the advection-dominated setting $\alpha \ll 1$. In this next section, we will consider the performance of the DLF and KF under different dynamic conditions, i.e., when $\alpha \approx 1$ or larger. The DLF

relies on the propagation of observations along characteristics determined by the advective part of the system. The evolution of these observations along these characteristics occurs in the presence of diffusion, which is determined by the derivatives of the imperfect model v. Thus, we expect diminishing returns in the high α regime for the DLF. We will examine this next. In the following experiments, $I = 20$, $\sigma = 1$, $\theta = \frac{1}{2}$, and the noise remains spatially uncorrelated, i.e., $\tilde{A} = 0$. The resulting average metrics as a function of α on a range of $\alpha \in [0.0001, 5]$ are depicted in Figure 4.15. As noted previously, the DLF has an advantage over the KF in all four analyzed metrics as long as advection dominates, i.e., $\alpha \ll 1$. However, this advantage decreases as α increases. Nonetheless, the DLF remains useful if the data is sparse.

In the advection-dominated case, the advantages of the DLF were more pronounced in cases of ill-informed models. Thus, as a last experiment, we will investigate if these advantages can be maintained as α increases. To this end, we assume uncertainties in the initial data amplitude and phase, i.e., $\sigma \sim \mathcal{U}[\frac{1}{2}, \frac{3}{2}]$ and $\theta \sim \mathcal{U}[0, 1]$. Further, we reintroduce the systematic uncertainty in phase speed into the model, i.e., $\tilde{A} = 1$, when simulating the truth. The number of observations is set to $I = 20$, while $\alpha \in [0.0001, 5]$. The resulting average metrics are depicted in Figure 4.16. We see now that the DLF, again, performs better in all four metrics over the entire range of analyzed α. For RMS and Mass error, as well as Calibration, the distance between DLF and KF decreases as α increases, while the advantage in terms of CoM error seems to be nearly unaffected by the values of α.

4.8 Discussion and Conclusions

The DLF approach to data assimilation was developed to handle hyperbolic (wave) dynamics. In this work, we extend the DLF approach to handle advection-diffusion dynamics. The DLF approach was first proposed in [13] and made operational in [5] on hyperbolic problems. A significant challenge in extending DLF to advection-diffusion problems is that the diffusion term needs to be evaluated along characteristic paths. We used the derivatives of the model as an estimator for this term. However, there are other alternatives, depending on the physics underlying the problem.

In this study, the evolution of observations through time was driven by a combination of data and the model at hand. Since the grid points used in model calculations and the characteristics along which observations travel in the Lagrangian frame did not necessarily coincide, extrapolation of the model to these off-grid points was required.

There are two ways the DLF can be formulated. In both cases, phase information is conveyed via the dynamic likelihood. In the multi-analysis case, observations or their projections forward in time, along with their uncertainties, can improve the extent of space in which observations affect the analysis product. In practice, the decision of which observations to keep for how long after their original measurement will need to balance computational complexity against improved accuracy.

The DLF approach requires a code that can solve the characteristics problem. With this solver, all additional implementation steps are no harder to implement than the classic KF. Like the KF, the computational complexity of the DLF approach applied to the KF is cubic in the maximum number of data points used at a time step. It can, however, still be significantly higher than the KF, since the DLF would be potentially applied more frequently than if applied at observation times exclusively. Some countermeasures to keep the DLF's complexity in check were discussed. Improved estimates make the higher computational cost justifiable.

Using numerical simulations, we demonstrated that the dynamic likelihood filter (DLF) outperformed the standard KF estimation concerning several metrics of accuracy. We showed that the DLF is superior to the KF when advection dominates diffusion, and observations are sparse and have high precision. Further, we demonstrated that:

- The DLF leads to a more accurate prediction of the truth than the KF, as demonstrated through its lower RMS in all experiments.

- The DLF estimates are significantly less sensitive to uncertainties in the initial data than the KF. It manages to predict the correct phase and amplitude within a shorter time and does so more accurately. As a result of capturing the phase more accurately, the center of mass of the solution is predicted with more accuracy. Further, the DLF gives more accurate local estimates (RMS) and predictions of overall mass.

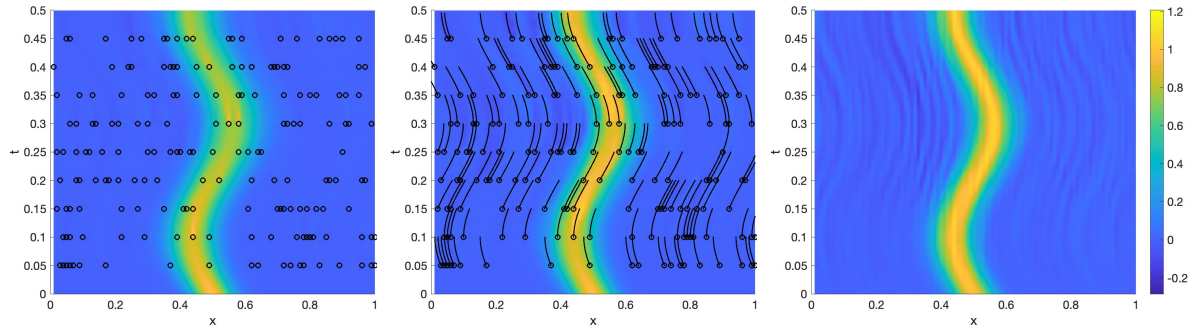
- The DLF leads to more accurate predictions even when the phase speed is affected by a great deal of uncertainty.
- The DLF is superior in estimating the variance, particularly when uncertainty beyond noise is introduced through uncertain initial data or an ill-informed model.

The last three points remain true, even when diffusion and advection are roughly the same ($\alpha \approx 1$).

In summary, the DLF approach to data assimilation on advection-diffusion problems shows great promise as an estimator, particularly when the observation network is sparse yet of low noise. The implementation requires special time integrators, but its computational overhead is well offset by producing better estimates. In [5], we showed that the DLF permits Bayesian estimates of model and pseudo-observations *into the future*, possibly beyond the present time when no observations are available. Using conventional data assimilation, forecasts will use the model-informed prior in the estimate of future moments. If the pseudo-observations inform a likelihood that is more compact than the prior, the forecast of the mean of the state may well be significantly different than the mean predicted via the prior only. This unique capability of being able to make Bayesian forecasts persists in the DLF approach, as applied to advection-diffusion problems.

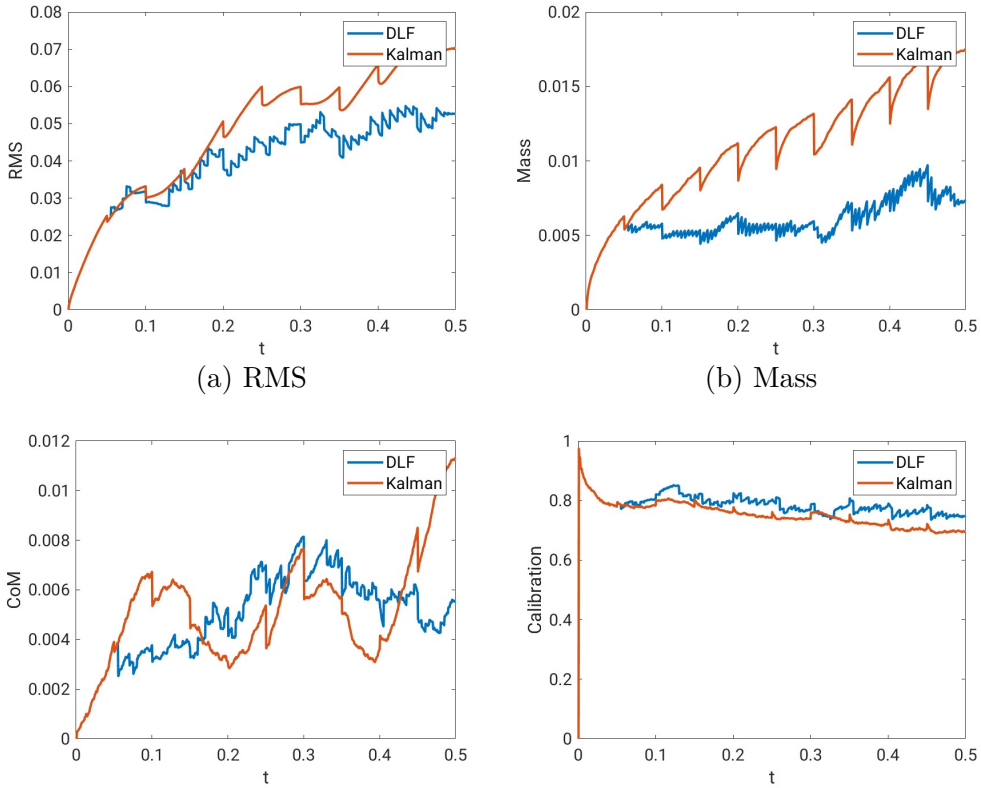
4.9 Acknowledgments

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.



(a) KF (b) DLF (c) Truth

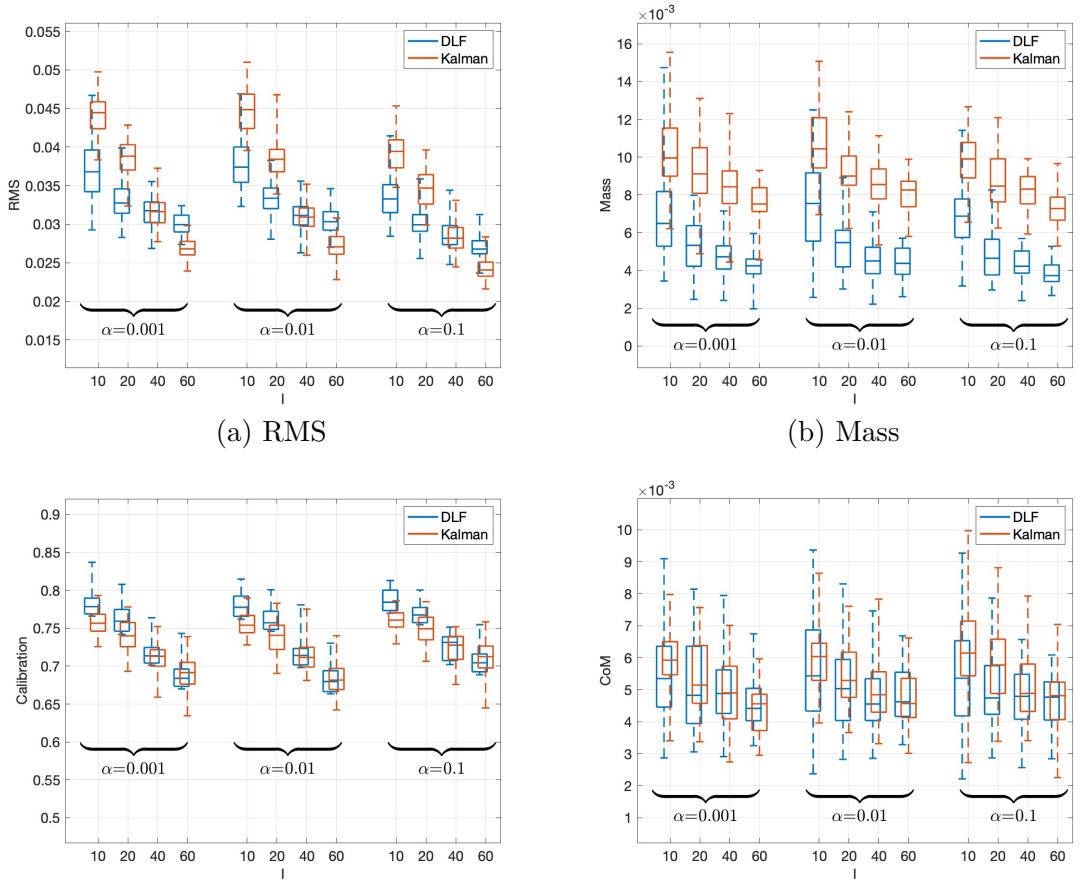
Figure 4.3: Posterior mean prediction as estimated by (a) the KF, and (b) the DLF, compared to (c) the truth. Advection dominated case, with $\alpha = 0.01$, initial data $\sigma = 1$ and $\theta = 0.5$ and spatially uncorrelated wave noise $A = 0.05$ and $\tilde{A} = 0$. Both filters use $I = 20$ observations per observation time. The locations of observations are randomly selected grid points, marked by black rings. Observation times are $T_O = \{0.05, 0.1, ..., 0.45\}$. The trajectories of pseudo-observations are shown as black lines. (In text mentions: p.119)



(a) RMS (b) Mass

(c) CoM (d) Calibration

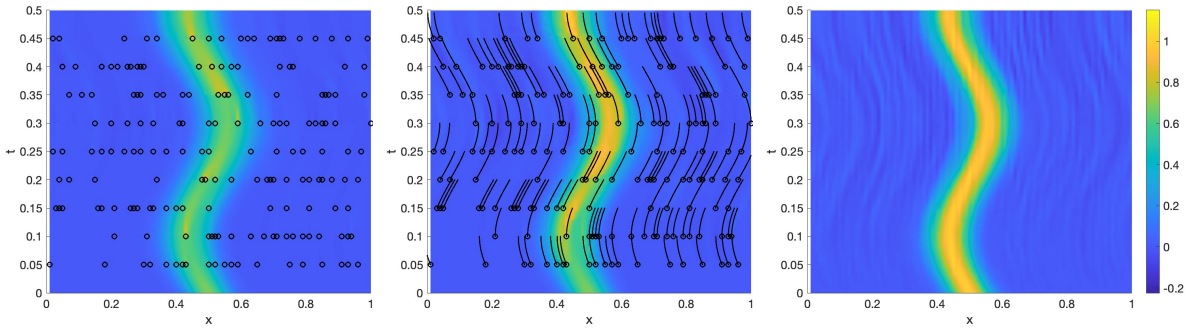
Figure 4.4: Time series of (a) RMS, (b) Mass, (c) CoM errors and (d) Calibration of the KF (red), the DLF (blue), averaged over 50 runs. Depicted are results from advection dominated cases with $\alpha = 0.01$, known initial data $\sigma = 1$ and $\theta = 0.5$ and spatially uncorrelated wave noise $A = 0.05$ and $\tilde{A} = 0$. Both filters use $I = 20$ observations per observation time. The locations of observations are randomly selected grid points. Observation times are $T_O = \{0.05, 0.1, ..., 0.45\}$. (In text mentions: p.119)



(a) RMS (b) Mass

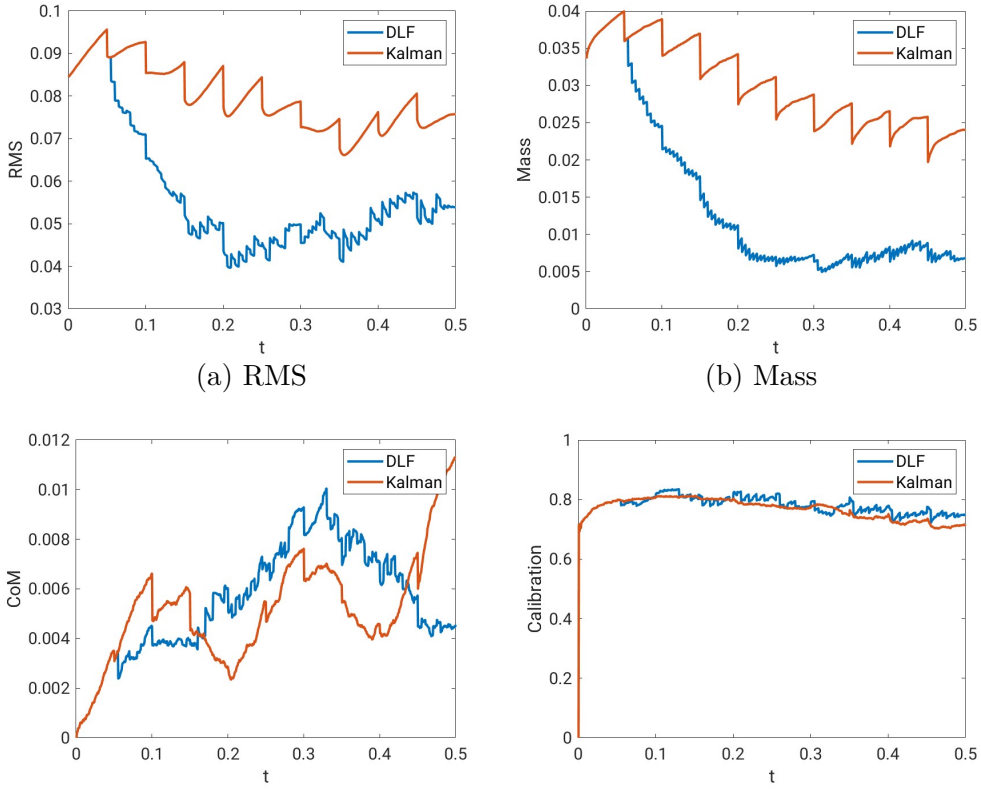
(c) CoM (d) Calibration

Figure 4.5: Average (a) RMS, (b) Mass, (c) CoM errors and (d) Calibration of KF (blue), and DLF (red), across 50 runs for spatially independent phase speed noise $A = 0.05, \tilde{A} = 0$, varying diffusion $\alpha =\in \{0, 0.001, 0.01\}$) and observations at $I = 10, 20$ and 40 random locations at every observation time $T_O = \{0.05, 0.1, ..., 0.45\}$. Initial data is assumed known with $\sigma = 1$ and $\theta = \frac{1}{2}$. (In text mentions: p.120)



(a) KF (b) DLF (c) Truth

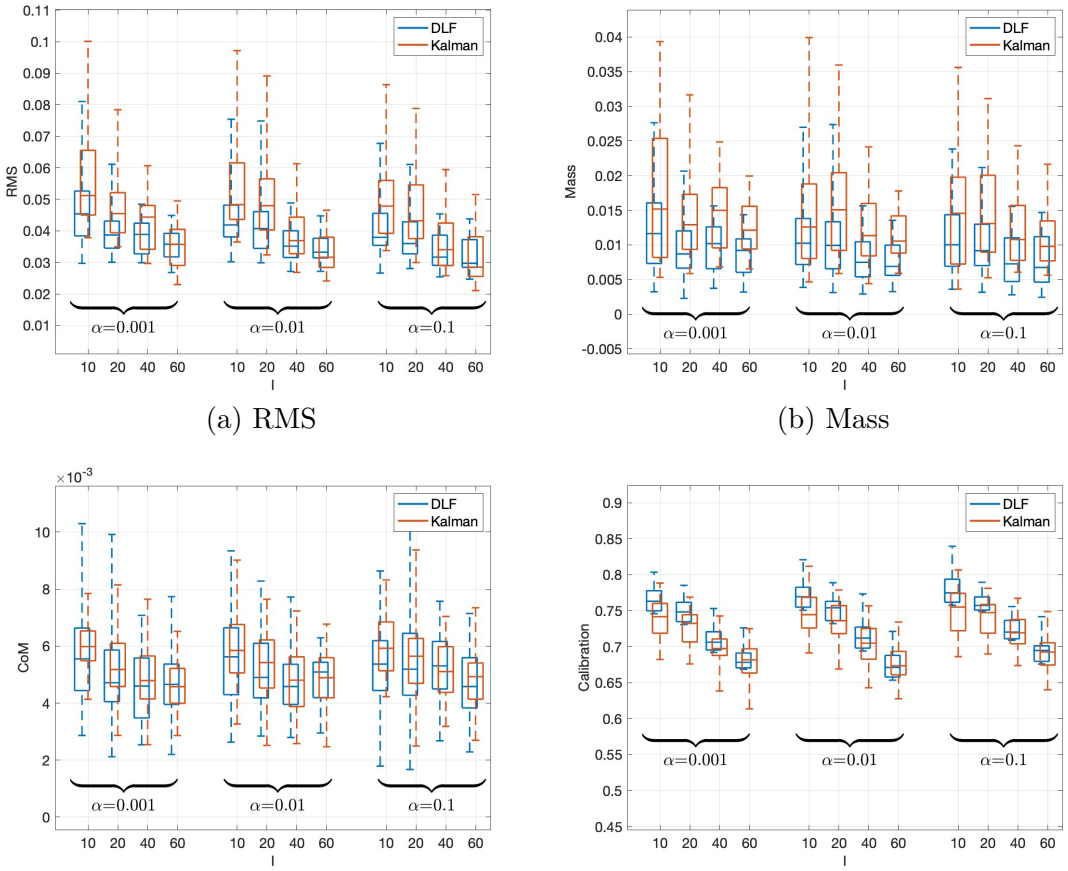
Figure 4.6: Posterior mean prediction as estimated by the (a) KF, and the (b) DLF, compared to the (c) truth. Advection dominated case with $\alpha = 0.01$, known initial phase $\theta = 0.5$ and spatially uncorrelated wave noise $A = 0.05$ and $\tilde{A} = 0$. The models use an incorrect initial amplitude of $\sigma = 0.7$ as opposed to the initial amplitude of the truth $\sigma = 1$. Both filters use $I = 20$ observations per observation time. The locations of observations are randomly selected grid points, marked by black rings. Observation times are $T_O = \{0.05, 0.1, ..., 0.45\}$. The trajectories of pseudo-observations are shown as black lines. (In text mentions: p.121)



(a) RMS (b) Mass

(c) CoM (d) Calibration

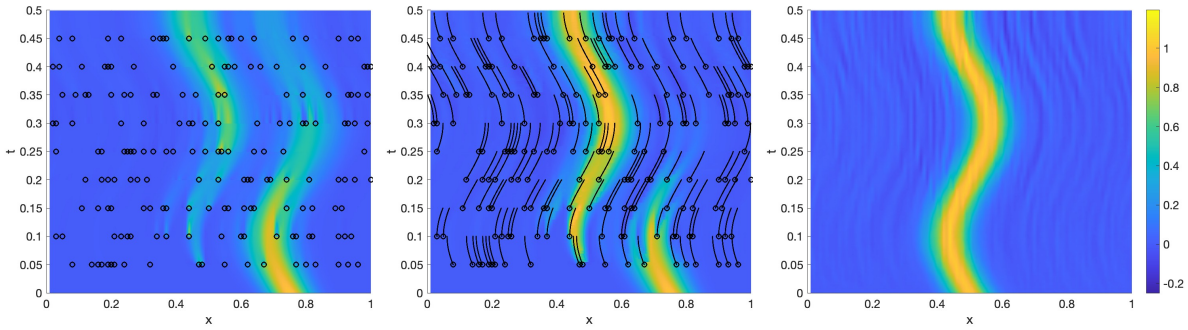
Figure 4.7: Time series of (a) RMS, (b) Mass, (c) CoM errors and (d) Calibration of the KF (red), the DLF (blue), averaged over 50 runs. Depicted are results from advection dominated cases with $\alpha = 0.01$, known initial phase $\theta = 0.5$ and spatially uncorrelated wave noise $A = 0.05$ and $\tilde{A} = 0$. The models use an incorrect initial amplitude of $\sigma = 0.7$ as opposed to the initial amplitude of the truth $\sigma = 1$. Both filters use $I = 20$ observations per observation time. The locations of observations are randomly selected grid points. Observation times are $T_O = \{0.05, 0.1, ..., 0.45\}$. (In text mentions: p.121)



(a) RMS (b) Mass

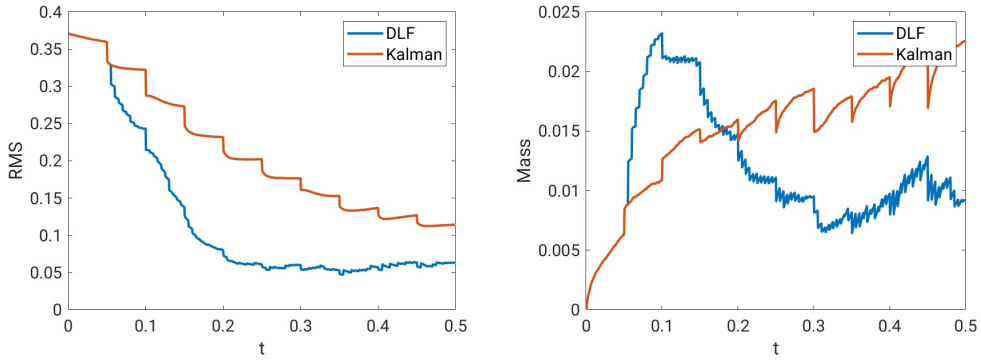
(c) CoM (d) Calibration

Figure 4.8: Average (a) RMS, (b) Mass, (c) CoM errors and (d) Calibration of KF (blue), DLF (red), across 50 runs for spatially independent phase speed noise $A = 0.05, \tilde{A} = 0$, varying difusion $\alpha = \in \{0, 0.001, 0.01\}$) and observations at $I = 10, 20$ and 40 random locations at every observation time $T_O = \{0.05, 0.1, ..., 0.45\}$. Initial amplitude is assumed uncertain with $\sigma = \mathcal{U}\left(\frac{1}{2}, \frac{3}{2}\right)$ and $\theta = \frac{1}{2}$. (In text mentions: p.121)



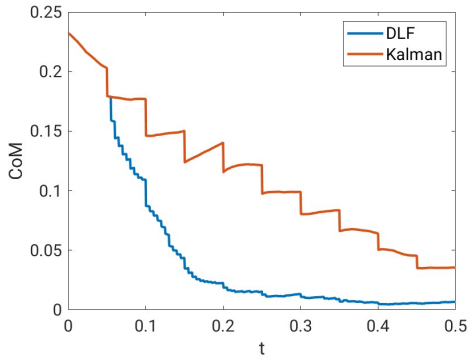
(a) KF (b) DLF (c) Truth

Figure 4.9: Posterior mean prediction as estimated by the (a) KF, and the (b) DLF, compared to the (c) truth. Advection dominated case with $\alpha = 0.01$, known initial amplitude $\sigma = 1$ and spatially uncorrelated wave noise $A = 0.05$ and $\tilde{A} = 0$. The models use an incorrect initial phase of $\theta = 0.25$ as opposed to the initial amplitude of the truth $\theta = 0.5$. Both filters use $I = 20$ observations per observation time. The locations of observations are randomly selected grid points, marked by black rings. Observation times are $T_O = \{0.05, 0.1, ..., 0.45\}$. The trajectories of pseudo-observations are shown as black lines. (In text mentions: p.122)

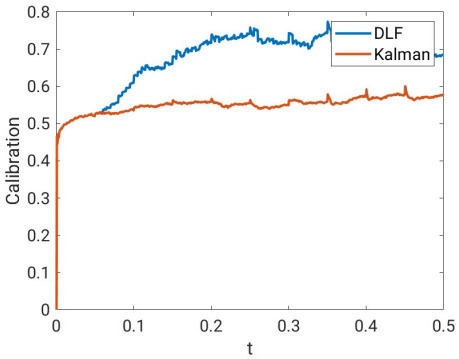


(a) RMS

(b) Mass

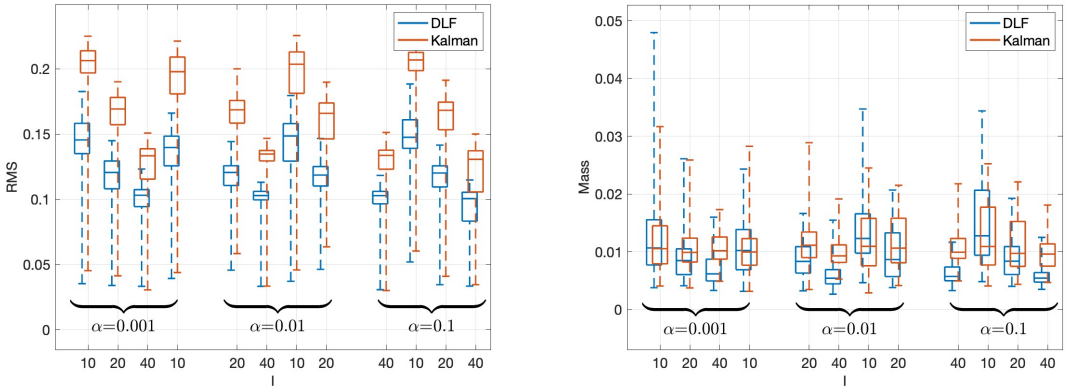


(c) CoM



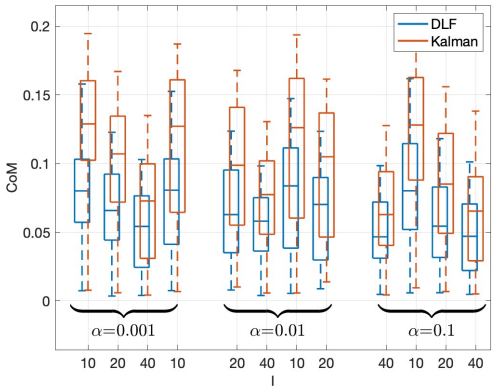
(d) Calibration

Figure 4.10: Time series of (a) RMS, (b) Mass, (c) CoM errors and (d) Calibration of the KF (red), the DLF (blue), averaged over 50 runs. Advection dominated cases with $\alpha = 0.01$, known initial amplitude $\sigma = 1$ and spatially uncorrelated wave noise $A = 0.05$ and $\tilde{A} = 0$. The models use an incorrect initial phase of $\theta = 0.25$ as opposed to the initial amplitude of the truth $\theta = 0.5$. Both filters use $I = 20$ observations per observation time. The locations of observations are randomly selected grid points. Observation times are $T_O = \{0.05, 0.1, ..., 0.45\}$. (In text mentions: p.122)

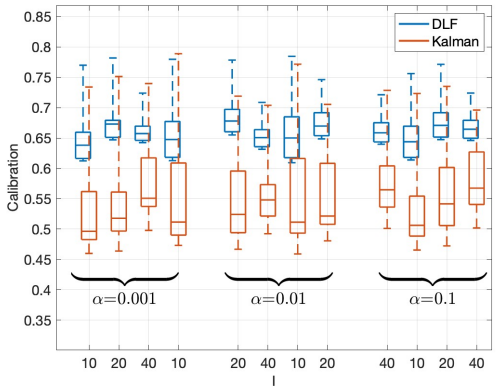


(a) RMS

(b) Mass

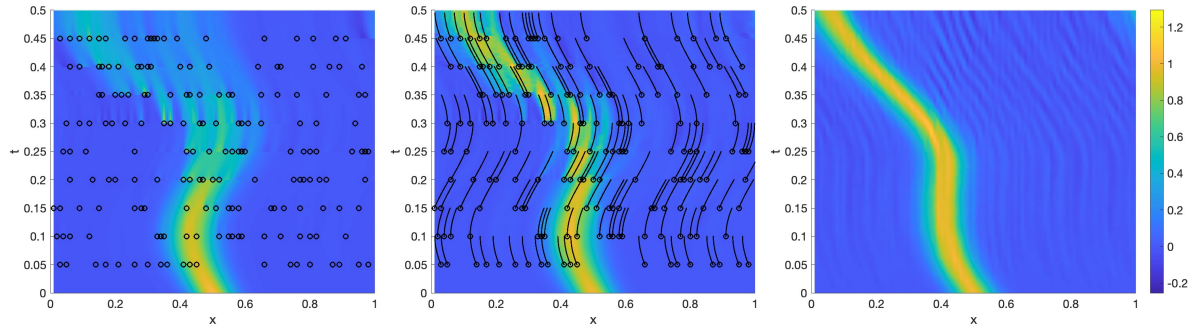


(c) CoM



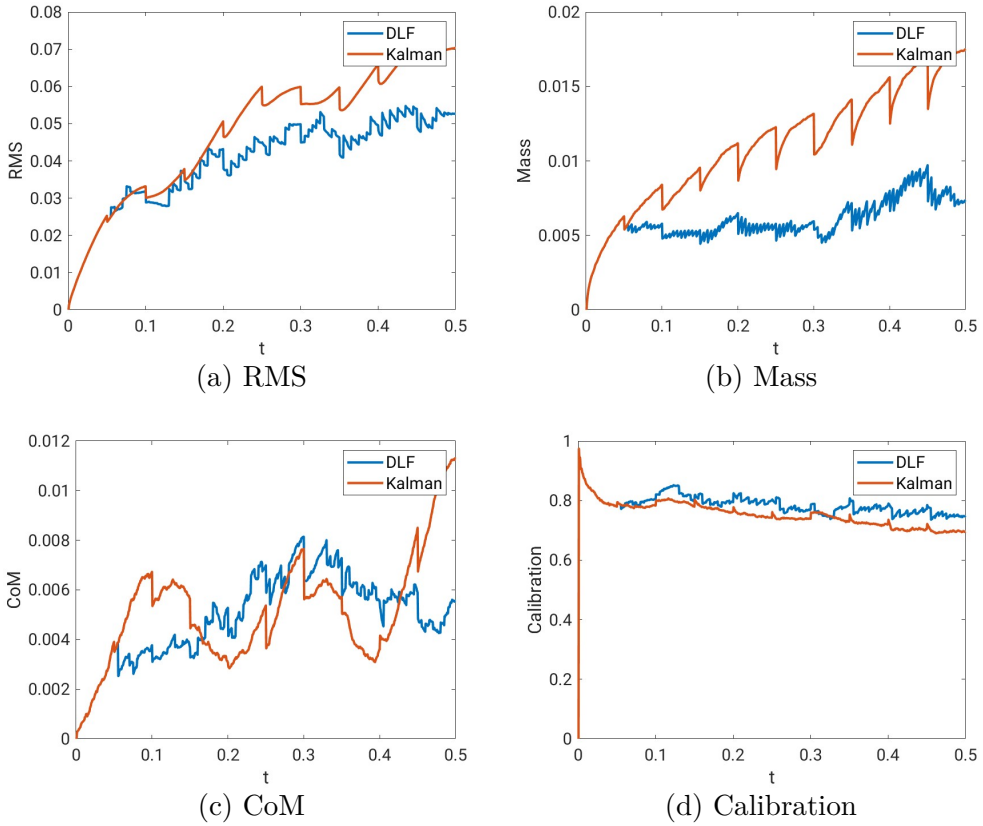
(d) Calibration

Figure 4.11: Average (a) RMS, (b) Mass, (c) CoM errors and (d) Calibration of KF (blue), DLF (red), across 50 runs for spatially independent phase speed noise $A = 0.05, \tilde{A} = 0$, varying difusion $\alpha = \in \{0, 0.001, 0.01\}$) and observations at $I = 10, 20$ and 40 random locations at every observation time $T_O = \{0.05, 0.1, ..., 0.45\}$. Initial phase is assumed uncertain with $\sigma = 1$ and $\theta = \mathcal{U}(0, 1)$. (In text mentions: p.122)



(a) KF (b) DLF (c) Truth

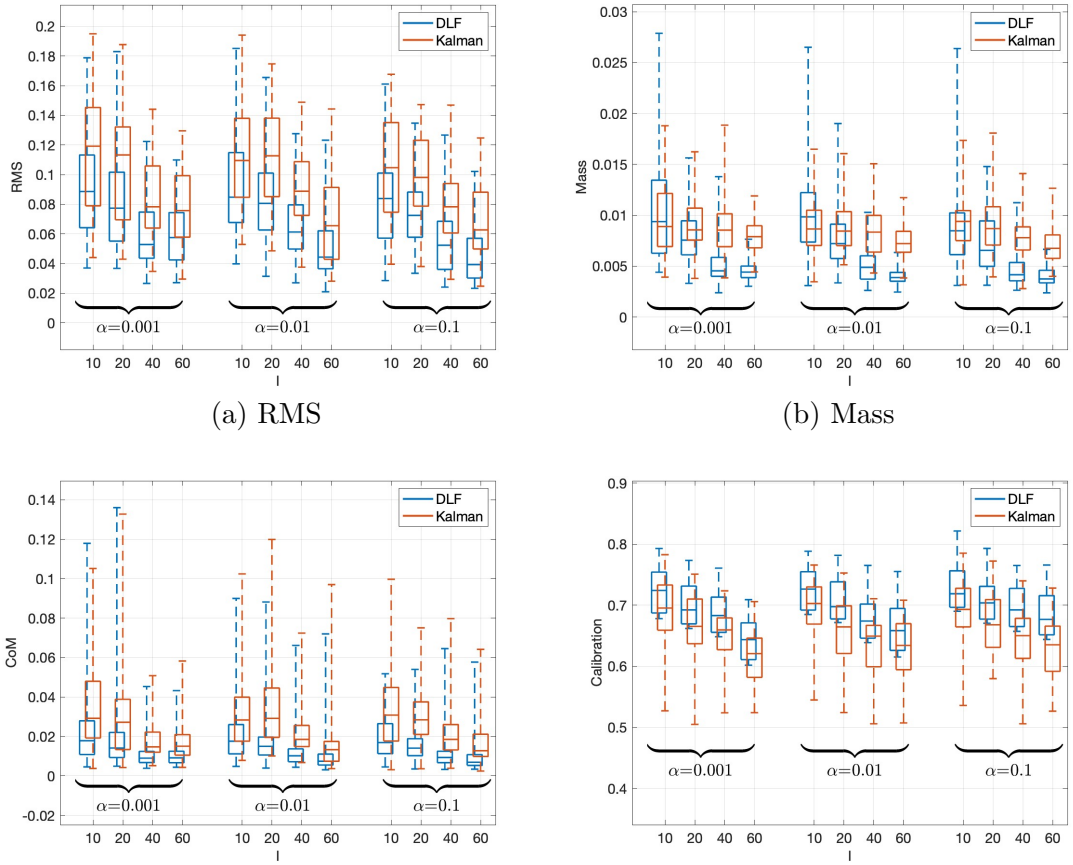
Figure 4.12: Posterior mean prediction as estimated by the (a) KF, and the (b) DLF, compared to (c) the truth. Advection dominated case with $\alpha = 0.01$, known initial data $\sigma = 1$ and $\theta = 0.5$. Phase speed noise of the truth is assumed spatially correlated $\tilde{A} = 1$, while both models assume $\tilde{A} = 0$, causing significant discrepancies in phase speed between truth and model. Both filters use $I = 20$ observations per observation time. The locations of observations are randomly selected grid points, marked by black rings. Observation times are $T_O = \{0.05, 0.1, ..., 0.45\}$. The trajectories of pseudo-observations are shown as black lines. (In text mentions: p.123)



(a) RMS (b) Mass

(c) CoM (d) Calibration

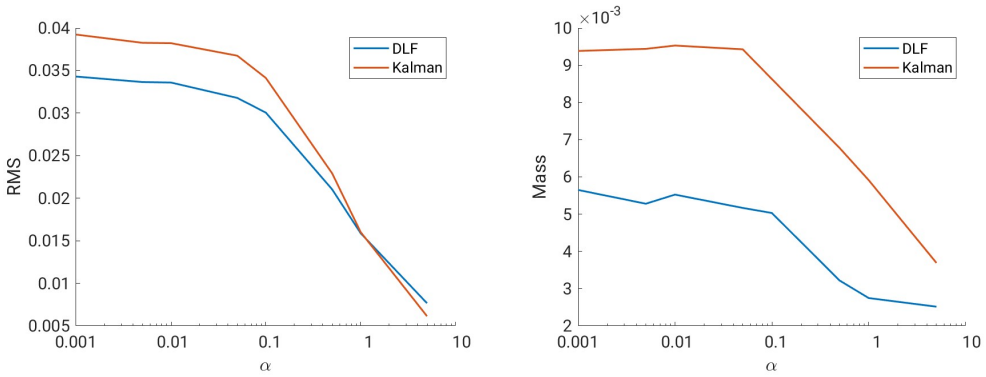
Figure 4.13: Posterior mean prediction as estimated by the (a) KF, the (b) DLF, compared to (c) the truth. Advection dominated case with $\alpha = 0.01$, known initial data $\sigma = 1$ and $\theta = 0.5$. Phase speed noise of the truth is assumed spatially correlated $\tilde{A} = 1$, while both models assume $\tilde{A} = 0$, causing significant discrepancies in phase speed between truth and model. Both filters use $I = 20$ observations per observation time. The locations of observations are randomly selected grid points. Observation times are $T_O = \{0.05, 0.1, ..., 0.45\}$. (In text mentions: p.123)



(a) RMS (b) Mass

(c) CoM (d) Calibration

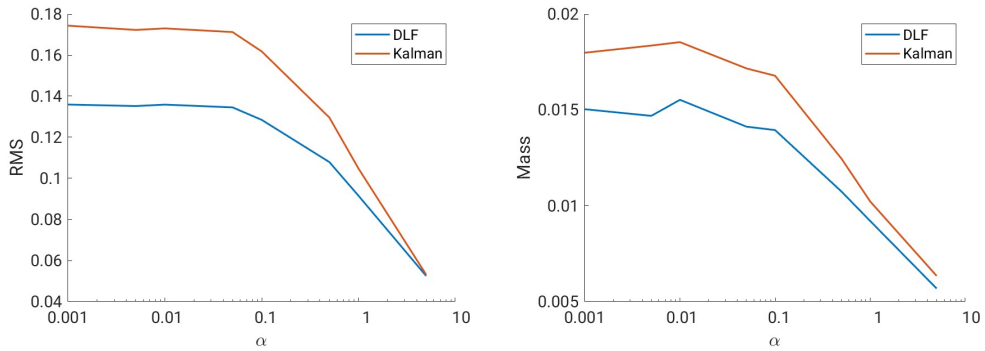
Figure 4.14: Average (a) RMS, (b) Mass, (c) CoM errors and (d) Calibration of KF (blue), DLF (red), across 50 runs for spatially correlated phase speed noise $A = 0.05, \tilde{A} = 1$, varying difusion $\alpha = \in \{0, 0.001, 0.01\}$) and observations at $I = 10, 20$ and 40 random locations at every observation time $T_O = \{0.05, 0.1, ..., 0.45\}$. Initial data is assumed known initial $\sigma = 1$ and $\theta = \frac{1}{2}$. (In text mentions: p.123)



(a) RMS (b) Mass

(c) CoM (d) Calibration

Figure 4.15: Average (a) RMS, (b) Mass, (c) CoM errors, and (d) Calibration of KF (blue) and DLF (red) as a function of α, across 50 runs for spatially independent phase speed noise $A = 0.05$, $\tilde{A} = 0$, and $I = 20$ randomly located observations available at every observation time $T_O = \{0.05, 0.1, ..., 0.45\}$ with known initial data $\sigma = 1$ and $\theta = \frac{1}{2}$. (In text mentions: p.124)



(a) RMS (b) Mass

(c) CoM (d) Calibration

Figure 4.16: Average (a) RMS, (b) Mass, (c) CoM errors and (d) Calibration of KF (blue), DLF (red), as a function of α, across 50 runs for spatially correlated phase speed noise ($A = 0.05, \tilde{A} = 1$) and $I = 20$ randomly located observations at every observation time $T_O = \{0.05, 0.1, ..., 0.45\}$ with noisy initial data $\sigma \sim \mathcal{U}[\frac{1}{2}, \frac{3}{2}]$ and $\theta \sim \mathcal{U}[0, 1]$ (In text mentions: p.124)

References

[1] F. J. Alexander, G. L. Eyink, and J. M. Restrepo. Accelerated Monte-Carlo for optimal estimation of time series. *Journal of Statistical Physics*, 119:1331–1345, 2005. 100

[2] A.J. Chorin, M. Morzfeld, and X. Tu. Implicit particle filters for data assimilation. *Communications in Applied Mathematics and Computational Science*, 5(2):221–240, 2010. 100

[3] G Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99:10143, 1994. 100

[4] G Evensen. Sampling strategies and square root analysis schemes for the EnKF. *Ocean Dynamics*, 54(6):539–560, 2004. 100

[5] Dallas Foster and Juan M. Restrepo. An improved framework for the Dynamic Likelihood Filtering approach to data assimilation. *Chaos*, 32(053118), 2022. 101, 102, 106, 112, 124, 126

[6] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. 100

[7] RE Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82:35–45, 1960. 100

[8] Johannes Krotz, Juan M. Restrepo, and Jorge M. Ramires. A likelihood approach to filtering for advection diffusion processes. Manuscript in preparation, 2024. 99

[9] H.J. Kushner. On the differential equations satisfied by conditional probability densities of Markov processes, with applications. *Journal of the Society for Industrial and Applied Mathematics Series A Control*, 2(1):106–119, 1962. 100

[10] Bruce A. McElhoe. An assessment of the navigation and course corrections for a manned flyby of mars or venus. *IEEE Transactions on Aerospace and Electronic Systems*, AES-2(4):613–623, 1966. 100

[11] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistics Society B*, 68:411–436, 2006. 100

[12] HE Rauch, F Tung, and CT Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3:1445–1450, 1965. 100

[13] J. M. Restrepo. Dynamic Likelihood Filter. *Quarterly Journal of the Royal Meteorological Society*, 2013. 100, 112, 124

[14] J. M. Restrepo and J. M. Ramírez. Calculating probability densities with homotoopy, and applications to particle filters. *International Journal of Uncertainty Quantification*, 12:71–89, 2022. 100

[15] S. Rosenthal, S. Venkataramani, A. Mariano, and J. M. Restrepo. Displacement data assimilation. *Journal of Computational Physics*, 330:594–614, 2017. 100

[16] Andreas Rossler. Second order Runge-Kutta methods for ito stochastic differential equations. *SIAM Journal on Numerical Analysis*, 47(3), 2009. 116

[17] Simo Sarkka. *Bayesian Filtering and Smoothing*. Cambridge University Press, Cambridge, 2013. 100

[18] G. L. Smith, S. F. Schmidt, and L. A. McGee. Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle. Technical report, National Aeronautics and Space Administration, 1962. 100

[19] Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968. 116

Vita

Johannes Krotz was born and grew up in Southern Germany. After high school, he attended the University of Konstanz, where he earned his Bachelor of Science in Physics in 2015. During his studies, he embarked on a six-month research exchange at Reykjavik University, collaborating on his thesis with Professor Sigurður Erlingsson and Professor Wolfgang Belzig. Following this, he continued his academic journey at the University of Konstanz, simultaneously pursuing a Bachelor of Science in Mathematics and a Master of Science in Physics. He completed the former in 2018 under the guidance of Professor Oliver Schn"urer and the latter in 2019 under the supervision of Professor Peter Nielaba.

In 2019, Johannes commenced his Doctor of Philosophy in Mathematics at Oregon State University, where he obtained a Master of Science in Mathematics in 2021. He later transferred to the University of Tennessee Knoxville, under the mentorship of Professor Juan Restrepo. During the period from 2020 to 2023, Johannes served as a research intern/student at both Los Alamos National Laboratory and Oak Ridge National Laboratory. At the time of submitting this dissertation, Johannes is expected to graduate from the University of Tennessee Knoxville with a Master of Science in Statistics and a Doctorate in Mathematics.

His research interests primarily revolve around numerical analysis, computational and statistical physics, and scientific computing. Following graduation, he plans to work as a Postdoctoral Researcher at Notre Dame University in collaboration with the Center for Exascale Monte Carlo Neutron Transport.