Flexible Monte Carlo Methods for Fluid and Light Transport Simulations

Damien Rioux-Lavoie

Department of Electrical and Computer Engineering

McGill University, Montreal

August, 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

Abstract

The remarkable progress in graphics processing units over recent decades has given rise to a significant increase in the application of computer graphics across various industries, such as in cinema and video games. Consequently, there has been a surge in opportunities to push the boundaries of computer graphics algorithms, leading to the development of highly intricate fluid simulations and photorealistic rendering. These methods now serve as the foundation for AAA games, animation features, and blockbuster movies.

This thesis emphasizes the development and application of flexible Monte Carlo techniques for two specific subfields within computer graphics, namely physically-based rendering and fluid simulations. The rationale behind this focus is the shared similarities between the equations and mathematical models governing both domains, presenting an opportunity to bridge the gap between them and formulate new and effective methods. Moreover, since the realism of a fluid simulation rendering is inherently tied to the simulation itself, and vice versa, advancements in both areas are crucial for achieving maximum impact.

First, we present a versatile two-stage mutation strategy based on the delayed rejection Markov chain Monte Carlo framework to generalize the Metropolis light transport algorithm. By generating multiple proposals informed by previous failures while maintaining Markov chain ergodicity, we can develop efficient strategies such as trying a cheap mutation first, followed by a more expensive one only upon failure. This

approach allows for the optimal allocation of computational resources, which is critical when tackling complex scenes.

Drawing on the success of Monte Carlo methods in physically-based rendering and, more recently, in discrete geometry processing, we propose a Monte Carlo approach for fluid simulations. Specifically, we employ the Feynman–Kac stochastic representation of the vorticity transport equation and devise a recursive Monte Carlo estimator of the Biot-Savart law that is able to generate pointwise approximate solutions. We expand this method with a stream function formulation that enables us to manage free-slip boundary conditions using a Walk-on-Spheres algorithm. To our knowledge, this is the first time that Monte Carlo methods have been studied in the context of fluid simulations, opening the door to a new family of solvers. We offer an in-depth examination of several potential directions for future research based on this novel numerical simulation modality.

Keywords: Computer Graphics, Physically-Based Light Transport Simulations, Fluid Simulations, Monte Carlo Integration, Markov Chain Monte Carlo Integration, Stochastic Process, Walk-on-Spheres Algorithm.

Résumé

Les progrès remarquables réalisés dans les unités de traitement graphique au cours des dernières décennies ont donné lieu à une augmentation significative de l'application de l'informatique graphique dans diverses industries, telles que le cinéma et les jeux vidéo. Par conséquent, il y a eu une augmentation des opportunités pour repousser les limites des algorithmes d'informatique graphique, conduisant au développement de méthodes de simulations de fluides très complexes et de rendus photoréalistes. Ces méthodes sont désormais à la base des jeux AAA, des longs métrages d'animation et dans les effets speciaux de films.

Cette thèse met l'accent sur le développement et l'application de nouvelles techniques de Monte Carlo flexibles pour deux sous-domaines spécifiques de l'informatique graphique, à savoir le rendu basé sur la physique et les simulations de fluides. La raison de ce choix est la similitude entre les équations et les modèles mathématiques régissant ces deux domaines, offrant ainsi l'opportunité de combler l'écart entre eux et de créer de nouvelles méthodes. De plus, étant donné que le réalisme d'un rendu de simulation de fluides est intrinsèquement lié à la simulation elle-même, et vice versa, il est essentiel de faire progresser les deux domaines pour obtenir un impact maximal.

Tout d'abord, nous présentons une stratégie de mutation en deux étapes flexible basée sur le cadre de la méthode de Monte Carlo à chaîne de Markov à rejet différé pour généraliser l'algorithme de transport de lumière de Metropolis. En générant plusieurs

propositions qui tiennent compte des échecs précédents tout en maintenant l'ergodicité de la chaîne de Markov, nous pouvons développer des stratégies efficaces telles que tenter d'abord une mutation économique, suivie d'une mutation plus coûteuse uniquement en cas d'échec. Cette approche permet une allocation optimale des ressources de calcul, ce qui est essentiel pour aborder des scènes complexes.

S'appuyant sur le succès des méthodes de Monte Carlo dans le rendu basé sur la physique et, plus récemment, dans le traitement de la géométrie discrète, nous proposons une approche de Monte Carlo pour les simulations de fluides. Plus précisément, nous utilisons la représentation stochastique de Feynman–Kac de l'équation de transport de vorticité et concevons un estimateur récursif de Monte Carlo de la loi de Biot–Savart qui est capable de générer des solutions approximatives ponctuelles. Nous étendons cette méthode avec une formulation de fonction de courant qui nous permet de gérer les conditions aux limites de glissement libre à l'aide d'un algorithme *Walk-on-Spheres*. À notre connaissance, c'est la première fois que les méthodes de Monte Carlo ont été étudiées dans le contexte des simulations de fluides, ouvrant la porte à une nouvelle famille de solveurs. Nous offrons un examen approfondi de plusieurs directions potentielles pour les travaux futurs basés sur cette nouvelle modalité de simulation numérique.

Mots-clés: : Infographie, Simulations de transport de lumière basées sur la physique, Simulations de fluides, Intégration de Monte Carlo, Intégration de Monte Carlo par chaîne de Markov, Processus stochastique, Algorithme de *walk-on-spheres*.

Acknowledgements

J'aimerais d'abord remercier mes parents, Marianne Rioux et Christian Bédard, pour leur soutien, leur encouragement et leur appui. Sans leur aide, je n'aurais jamais pu atteindre cette étape universitaire importante. Je remercie également ma conjointe, Laura Cheron-Leboeuf, pour son amour, sa patience sans fin et sa résilience inspirante. Évidemment, je remercie mon superviseur, Derek Nowrouzezahrai, pour sa confiance et son soutien inconditionnel durant les moments les plus difficiles.

J'aimerais aussi prendre le temps de remercier toutes les personnes que j'ai eu la chance de rencontrer et de travailler avec au fil des ans : Adrien Gruson, Joey Litalien, Nicolas Vibert, Yangyang Zhao, Sayantan Datta, Toshiya Hachisuka, Christopher Batty, Jean-Philippe Guertin, Arnaud Schoentgen, Amir Hossein Rabbani, Ryusuke Sugimoto, Andrea Feder, Olivier Pomarez, ainsi que plusieurs autres... Votre aide m'a été précieuse.

Naturellement, je tiens aussi à remercier le comité d'évaluation d'avoir accepté d'évaluer ma thèse, le département de génie électrique et informatique de McGill (ECE) pour son soutien administratif et financier, ainsi que le conseil de recherches en sciences naturelles et en génie du Canada (NSERC).

Enfin, je dédie cette thèse à mon premier fils, Élie, qui m'a tant appris durant son court passage dans ce monde, à mon second fils Edgar pour m'avoir donné la date limite qu'il m'aura fallu pour terminer cette rédaction, ainsi qu'à mon chat Pootsy pour tout l'amour qu'elle a su me donner.

Table of Contents

List of Figures

List of Algorithms

List of Abbreviations

2D	two-dimensional
3D	three-dimensional
ACM	Association for Computing Machinery
AD	automatic differentiation
AMH	adaptive Metropolis–Hastings
APIC	affine particle-in-cell
BDPT	bidirectional path tracing
BS	Biot–Savart
BSDF	bidirectional scattering distribution function
BVH	bounding volume hierarchy
CDF	cumulative distribution function
CFD	computational fluid dynamics
CG	computer graphics
CGI	computer-generated imagery
CUDA	compute unified device architecture
CV	control variates
DA	delayed acceptance
DGP	discrete geometry processing
DR	delayed rejection

DRAM delayed rejection adaptive Metropolis

DRMLT delayed rejection Metropolis light transport

FBSDS forward-backward stochastic differential systems

FEM finite element methods

FK Feynman–Kac

FLIP fluid-implicit-particle

GDR generalized delayed rejection

GPU graphics processing units

H2MC Hessian Hamiltonian Monte Carlo

HMC Hamiltonian Monte Carlo

IID independent and identically distributed

IS importance sampling

KAMH kernel adaptive Metropolis–Hastings

LBM lattice Boltzmann methods

LLN law of large numbers

MAC marker-and-cell

MALA Metropolis adjusted Langevin algorithm

MC Monte Carlo

MCFluid Monte Carlo Method for Fluid Simulation

MCI Monte Carlo integration

MCMC Markov chain Monte Carlo

MH Metropolis–Hastings

MIS multiple importance sampling

MLT Metropolis light transport

MMALA manifold Metropolis adjusted Langevin algorithm

MMLT multiplexed Metropolis light transport

MPM material point method

MSE mean squared error

MTM multiple try Metropolis

NS Navier–Stokes

PBF position based fluid

PBFA physically based fluid animation

PBR physically based rendering

PDE partial differential equation

PDF probability density function

PIC particle-in-cell

PMF probability mass function

PSS primary sample space

PSSMLT primary sample space Metropolis light transport

QMC quasi-Monte Carlo

QN-MCMC quasi-Newton Markov chain Monte Carlo

RJMCMC reversible jump Markov chain Monte Carlo

RJMLT reversible jump Metropolis light transport

RK4 Runge–Kutta 4th order

RKHS reproducing kernel Hilbert space

rMSE relative mean squared error

RMSE root mean squared error

RV random variable

RWMH random walk Metropolis–Hastings

SDE stochastic differential equation

SIGGRAPH	Special Interest Group on Computer Graphics and Interactive Techniques
SIR	sampling importance resampling
SMC	sequential Monte Carlo
SPH	smoothed particles hydrodynamics
TOG	Transactions on Graphics
VFX	visual effects
WoS	Walk-on-Spheres

Chapter 1

Introduction

Over recent decades, computer graphics (CG), also called computer-generated imagery (CGI), has made significant advancements in producing highly realistic images, largely due to the enhanced computational capabilities of modern graphics processing units (GPU). As a result, CG now plays a pivotal role in the entertainment industry, facilitating the creation of lifelike visual effects (VFX) in movies, animated features, and AAA video games (See Figure 1.1). Key factors in achieving such realism include simulating the interaction between light and objects or materials within a scene to create photorealistic images and modeling fluid behaviors for added credibility and dynamics. These two tasks are substantial enough to warrant dedicated fields of study: physically based rendering (PBR), also called rendering and light transport, and physically based fluid animation (PBFA), commonly referred to as fluid simulation.

Specifically, PBR is a technique that involves converting a three-dimensional (3D) scene, which includes 3D objects, materials, volumes, lights, and a camera, into a two-dimensional (2D) image that accurately represents the way light interacts with objects and materials in the scene. PBR takes into account various properties of materials, such as diffuse reflection, specular reflection, and subsurface scattering, as

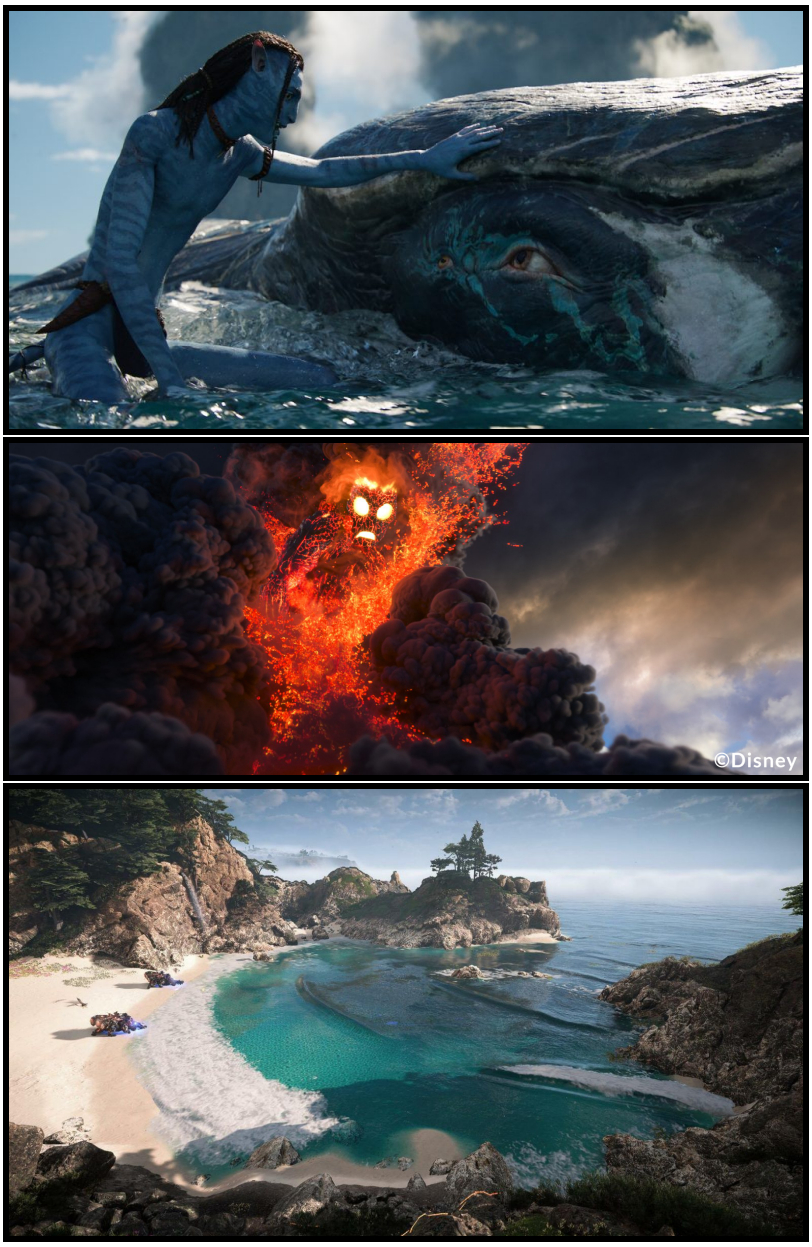


Figure 1.1: CGI in entertainment. Progress in rendering and fluid simulations within the entertainment sector: (Top) Blockbuster film *Avatar: The Way of Water* - Credit: 20th Century Studios, (middle) Animated movie *Moana* - Credit: Walt Disney Animation Studios, and (bottom) Video game *Horizon Forbidden West* - Credit: Sony Interactive Entertainment.

well as the physical properties of light, like absorption, reflection, and refraction, to create photorealistic images.

PBFA is based on computational fluid dynamics (CFD) and uses numerical methods and algorithms to accurately model fluid motion, encompassing phenomena such as smoke, liquid, and fire. These simulations consider various factors, including fluid viscosity, density, and external forces like gravity and pressure, to create realistic animations and effects. Compared to CFD, which aims at predicting the actual flow of real-live fluids, PBFA focuses on generating plausible visual effects. For this reason, PBFA often trades accuracy, predictability and good-enough approximations in favor for speed and simplicity.

Despite considerable advancements, the quest for perfect realism in CG remains a work in progress, with both PBR and PBFA still encountering numerous challenges. In the realm of PBR, Monte Carlo (MC) methods have largely outpaced traditional finite element methods (FEM), better addressing complex radiometric effects and increasingly intricate geometric and reflectance models. Nonetheless, the radiometric dynamics of light transport can become exceedingly complex, particularly in scenes featuring combinations of complex physically based materials and highly detailed geometry, giving rise to a host of numerical challenges.

MC rendering tackles the rendering equation by randomly sampling light paths and accumulating their contributions in the corresponding pixels. However, in scenes with highly glossy materials or with complex visibility, the probability of randomly sampling valid light paths that connect the camera to the light source approaches zero, causing the algorithm to struggle to converge to the correct image. Indeed, while these events can be rare, they can still contribute to a lot of energy in the image, like a caustic or backlit room for instance.

The Metropolis light transport (MLT) algorithm was developed to address this issue by mutating valid light paths toward regions of interest. However, designing efficient, robust, and general-purpose mutation strategies remains an unsolved problem. For instance, complex mutation strategies can be more effective at equal sample count but

often necessitate much more computation. For this reason naively using such mutation strategy over the whole image can result in very slow rendering considering that many parts of the image would bold well without it. On the other hand, completely abandoning these strategies might lead to under-exploration of essential regions within the scene, further complicating the pursuit of perfect realism. In summary, finding a one-size-fits-all mutation strategy that is efficient and effective is quite difficult and finding a way to do so would be very beneficial.

In recent years, a subfield of CG, called discrete geometry processing (DGP), has adopted MC techniques to overcome some of the limitations of traditional approaches. Previously, DGP relied heavily on FEM, which constrained the geometric complexity that could be handled by their algorithms. These methods were often unable to efficiently address challenges related to high-resolution meshes, sharp features, and intricate topologies in 3D models without suffering from extreme computational overhead, stability issues and algorithms complexity increases. The adoption of MC techniques in DGP has led to the development of new methods that demonstrate both geometric flexibility and robustness. These methods can handle complex shapes and topologies more efficiently, while still providing accurate and stable solutions even with few computational resources. This paradigm shift has effectively created a new area of research, expanding the possibilities for handling and manipulating 3D models in CG.

In contrast, PBFA has never received the MC treatment, continuing to rely on deterministic frameworks like FEM. As a result, PBFA often faces the same difficulties that PBR and DGP faced in the past when handling complex geometric boundaries and large domains. To address these challenges, PBFA requires careful domain discretization, which can involve adaptive mesh refinement techniques or specialized solvers to accurately represent the fluid behavior near complex geometries which comes at the cost of computational efficiency, robustness, stability and complexity. By embracing MC techniques, it is possible that PBFA could further advance, enabling more

efficient and accurate simulations in cases where fluid interactions with complex geometries are crucial. This could, in turn, lead to even more realistic and visually stunning effects in cinema, animations, and video games.

1.1 Objectives

Building on our earlier discussion, the primary aim of this Ph.D. thesis is to develop efficient and effective MC methods for CG applications. Specifically, we will concentrate on two subfields of CG: PBR and PBFA. This focus is motivated by two main factors. First, the equations and mathematical models underlying both fields exhibit similarities, offering a unique opportunity to bridge the gap between them and potentially develop innovative new methods. Second, the visual realism of a fluid simulation is intrinsically tied to the rendering quality, and vice versa, necessitating advancements in both fields to maximize their impact on achieving highly realistic CG.

In order to achieve this objective, our first focus will be on tackling the challenge of creating adaptable and efficient mutation strategies within MLT using the delayed rejection (DR) framework. We aim to devise a mutation strategy capable of adjusting to the diverse levels of complexity present in challenging scenes. By accomplishing this, we expect to develop rendering algorithms that surpass the current state of the art in terms of robustness and speed, ultimately reducing their computational requirements.

Next, we will investigate the potential for enhancing the robustness and flexibility of fluid simulations when dealing with complex geometries by incorporating MC methods. We anticipate that this exploration will have a similar impact on fluid simulations as it did on the fields of PBR and DGP when they adopted MC techniques. Ideally, this research will lead to the development of a new family of methods capable of accomplishing tasks that were previously unattainable using more traditional approaches, such as FEM.

1.2 Contribution to General Knowledge

Over the course of this Ph.D., I made significant contributions to the field of MC techniques in CG by authoring two research papers. Both of these papers were published in the prestigious Association for Computing Machinery (ACM) Transactions on Graphics (TOG), the top-tier journal in the realm of CG. Furthermore, each paper was chosen for presentation at the esteemed ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH) conferences held in North America and Asia. The following sections will detail the specific contributions of each paper.

1.2.1 Delayed Rejection Metropolis Light Transport

Damien Rioux-Lavoie, Joey Litalien, Adrien Gruson, Toshiya Hachisuka, and Derek Nowrouzezahrai. Delayed rejection Metropolis light transport. *ACM Transactions on Graphics (TOG)*, 39(3):1–14, 2020

In this paper, we generalized the MLT framework to employ a flexible two-stage mutation strategy based on DR Markov chain Monte Carlo (MCMC). Our approach generates multiple proposals based on the failure of previous ones, all while preserving Markov chain ergodicity. This allows us to reduce error while maintaining fast global exploration and low correlation across chains. Direct application of DR to MLT leads to low acceptance probabilities, and so we also proposed a novel transition kernel to alleviate this issue. In addition, we devised two light transport algorithms based on this framework, including bold-then-timid and cheap-then-expensive, that we showcased in three different application settings. Our techniques can be seamlessly integrated into any primary sample space (PSS) algorithm, delivering consistently improved outcomes across various challenging scenes. Chapter 4 is a partial reproduction of this paper. In summary, the contributions of this publication to original knowledge include:

- A significant enhancement to the MLT algorithm by introducing a two-stage proposal mechanism that is capable of adapting to local structures in target densities. This mechanism is based on the DR framework, which has not been previously applied in the field of CG. This innovation represents a substantial step forward in the development of advanced MC methods for rendering applications.
- A novel combination of transition kernels, called pairwise orbital mutation, to address the vanishing acceptance issue encountered in the original DR method. This innovative approach effectively mitigates the problem and is not restricted to light transport applications, making it a contribution to the broader field of MCMC methods.
- A waste recycling strategy that reuses all the samples generated during the DR stages, leveraging the method of expected values. This approach promotes efficient resource utilization and enhances the overall performance of the DR algorithm.
- Three distinct applications of the DR framework specifically for MLT, which automatically balances local exploration and computational efficiency. We benchmarked the versatility of these applications and showed that they improved convergence at equal time and with minimal implementation effort. This demonstrates the potential of our approach in handling diverse rendering challenges.
- Our work not only offers immediate improvements to rendering algorithms but also paves the way for numerous future research opportunities. By extending the application of the DR framework and developing a novel transition kernel, we have provided a foundation for further exploration and refinement in the field of MLT. This opens up new possibilities for both theoretical advancements and practical implementations in various rendering applications.

I collaborated closely with Joey Litalien as a co-primary investigator in this research. My contributions to this project encompass:

- Co-developing the theoretical framework in collaboration with Joey Litalien.
- Co-creating the waste recycling strategy alongside Joey Litalien.
- Developing the Kelemen-then-pairwise Orbital application.

- Developing the cheap-then-expensive application that utilizes the Hessian-Hamiltonian proposal.
- Actively participating in discussions and generating ideas for future work.
- Contributing to the implementations in both Mitsuba and DPT platforms.

Joey Litalien played a major role in writing, creating figures, executing experiments, and designing the multiplexed-then-subpath application. He also created the interactive viewer used in the project website and developed numerous scenes showcased in our results. Adrien Gruson contributed his extensive light transport expertise to assist with implementation, experimentation, and application development. The evaluation of the system was collaboratively designed and carried out by Joey Litalien, Adrien Gruson, and myself. Toshiya Hachisuka and Derek Nowrouzezahrai provided valuable guidance and contributed numerous ideas through their supervision. The manuscript was jointly authored by Joey Litalien, Adrien Gruson, Toshiya Hachisuka, Derek Nowrouzezahrai, and myself.

1.2.2 A Monte Carlo Method for Fluid Simulation

Damien Rioux-Lavoie, Ryusuke Sugimoto, Tümay Özdemir, Naoharu H Shimada, Christopher Batty, Derek Nowrouzezahrai, and Toshiya Hachisuka. A Monte Carlo method for fluid simulation. *ACM Transactions on Graphics (TOG)*, 41(6):1–16, 2022

In this second publication, we introduced a novel MC-based fluid simulation technique that enables pointwise and stochastic estimation of fluid motion. By utilizing the Feynman–Kac (FK) stochastic representation of the vorticity transport equation, we proposed a recursive MC estimator for the Biot–Savart (BS) law and expanded it with a stream function formulation, allowing us to address free-slip boundary conditions through a Walk-on-Spheres (WoS) algorithm. Inspired by MC literature in rendering, we developed and compared variance reduction schemes specifically tailored for fluid simulation contexts, demonstrating their applicability to intricate boundary settings. We

also outlined a simple and practical implementation featuring temporal grid caching. We validated the accuracy of our approach through quantitative and qualitative evaluations across various settings and domain geometries, while extensively examining the design space of our parameters. Furthermore, we offered an in-depth discussion on several avenues of future work stemming from this novel numerical simulation method. Chapter 5 is a partial reproduction this paper. In summary, the contributions of this publication to original knowledge include:

- A novel MC fluid solver that employs recursive, pointwise probabilistic solutions to the 2D Euler equations based on the BS law. This novel approach revolutionizes fluid simulation by offering a fundamentally different perspective on how to solve the underlying equations, paving the way for more diverse and adaptable techniques.
- A non-recursive solver that utilizes spatiotemporal caches to address the exponential complexity of the original method and achieve efficient fluid simulations.
- A novel WoS approach for addressing inflow, outflow, and free slip solid boundary conditions using stream functions. This contribution extends the applicability of MC methods in fluid simulation to a broader range of boundary conditions, further enhancing the versatility of this new technique.
- A generalized MC solver for the full 3D incompressible Navier–Stokes (NS) equations derived from the FK stochastic representation. To our knowledge, this is the first MC estimator for solutions of the general NS equations. This introduces an extensive range of new solvers with distinct properties compared to deterministic ones.
- The first application of MC variance reduction techniques in fluid simulation. By adapting these methods from the rendering domain to fluid simulation, we have demonstrated their potential for improving the efficiency and accuracy of fluid simulations, leading to more robust and reliable results in various applications.
- A comprehensive roadmap outlining open challenges and opportunities for the development of scalable MC fluid simulation methods. This exploration of future research directions enables the scientific community to build upon our work, securing the continued growth and evolution of MC-based fluid simulation techniques.

I collaborated closely with Ryusuke Sugimoto as a co-primary investigator in this research toward the end of this project. This is due to taking paternity leave during the final stages leading up to the submission deadline. My contributions to this project include:

- Developing and implementing the theoretical framework for the full NS application.
- Creating and implementing the importance sampling procedure.
- Designing and implementing the antithetic sampling procedure.
- Developing and implementing the moving and inflow/outflow boundary treatment.
- Contributing to the theoretical aspects of the control variate procedure.
- Participating in discussions and implementation of the free-slip boundary treatment.
- Contributing to the development of caching methods, including uniform grid and particle-based approaches.
- Assisting with the validation of the method, including comparisons, parameter studies, and convergence analysis.
- Actively participating in discussions and generating ideas for future work.
- Contributing to the full implementation of the method in C++.

Ryusuke Sugimoto reimplemented the codebase in compute unified device architecture (CUDA) to achieve interactive rates and also implemented the control variate approach, the vortex segment advection method, and conducted experiments with subgrid-sized and triangle soup obstacles. Tümay Özdemir contributed to the FLIP solver implementation and conducted comparisons against it. She also participated in the implementation and comparison of the vortex particle solver. Naoharu Shimada contributed to the free-slip boundary treatment development, the 2D implementation, and the adaptive grid caching approach. Toshiya Hachisuka was the first to propose the application of MC integration to the BS kernel and made significant overall contributions to the paper. Christopher Batty played a vital role in developing the free-slip boundary treatment, especially for the challenging 3D case. Derek Nowrouzezahrai contributed to numerous discussions, writing, and provided

supervision. Lastly, the manuscript was a joint effort of Ryusuke Sugimoto, Christopher Batty, Derek Nowrouzezahrai, Toshiya Hachisuka, and myself.

1.2.3 Compact Poisson Filters for Fast Fluid Simulation (not included in this thesis)

> Amir Hossein Rabbani, Jean-Philippe Guertin, Damien Rioux-Lavoie, Arnaud Schoentgen, Kaitai Tong, Alexandre Sirois-Vigneux, and Derek Nowrouzezahrai. Compact Poisson filters for fast fluid simulation. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022

Besides these two primary papers, I also contributed to an additional conference paper (SIGGRAPH North America 2022) during an internship with the environment team at Ubisoft La Forge. I have since become a full-time research and development scientist (RDS) with the same team. Although this paper developed a novel, fast, and high-fidelity fluid simulation method, its contributions were not related to MC methods. After careful consideration, **I decided to exclude this publication from my thesis** to focus exclusively on my principal works.

1.3 Outline of the Thesis

This thesis is organized into seven chapters, including the present introduction. Chapter 2 offers an extensive review of the relevant literature related to our work. This section is structured into three distinct parts: MC, PBR, and PBFA. Chapter 3 delivers a rigorous exposition of the background knowledge required to comprehend the methods presented in the following chapters. We start by discussing the basics of measure-theoretic probability theory, stochastic processes and then proceed to examine the various MC methods employed in this thesis. Next, we explore the fundamentals of

PBR, encompassing radiometric quantities, the rendering equation, the path integral, and MLT. Lastly, we cover the central concepts of PBFA, including the NS equations, the vorticity transport equation, the BS law and semi-Lagrangian advection. Chapter 4 presents a novel MLT algorithm based on the DR framework. We begin by contextualizing this work concerning the objectives of this thesis. Then, we proceed to develop our two-stage DR framework, showcase some of its applications in light transport and showcase our results. Chapter 5 presents a MC method for numerical fluid simulation. We first situate this work in relation to the goals of this thesis. Then, we present the core ideas behind our new framework in a simplified 2D setup, follow up with some advanced extensions and applications, and finish with a presentation of our results. In Chapter 6, we conduct a comprehensive scholarly discussion of all the results encountered in Chapter 4 and Chapter 5. This will include a discussion of the results, limitations of the methods, and future avenues of research. Furthermore, we will offer a glimpse into the other projects explored during this Ph.D. journey. While these projects remains unpublished, they represent valuable learning experiences and stepping stones towards our scientific progress. Finally, in Chapter 7, we provide a summary of the most significant contributions of our work and tie them back into our objectives.

Chapter 2

Literature Review

This thesis introduces two novel Monte Carlo (MC) applications within computer graphics (CG). The first method improves the efficiency and flexibility of Metropolis light transport (MLT), while the second is the first MC framework designed for fluid simulations. To provide context for our contributions, we offer a comprehensive literature review of the most pertinent works related to these subjects. This chapter will be divided into three distinct sections, addressing MC methods and the two specific CG fields covered in this thesis, namely physically based rendering (PBR) and physically based fluid animation (PBFA). This will help contextualize our contribution and demonstrate its significance.

2.1 Monte Carlo Methods

MC methods consist of a family of numerical methods that rely on random sampling to approximate the results of different types of problems. More specifically, they prove to be useful when the problem is high dimensional or proves to be difficult to express using other deterministic approaches, as they often require strong regularity assumptions. Common problems handled by these methods consist of optimization, numerical integration and sampling according to intractable probability distributions.

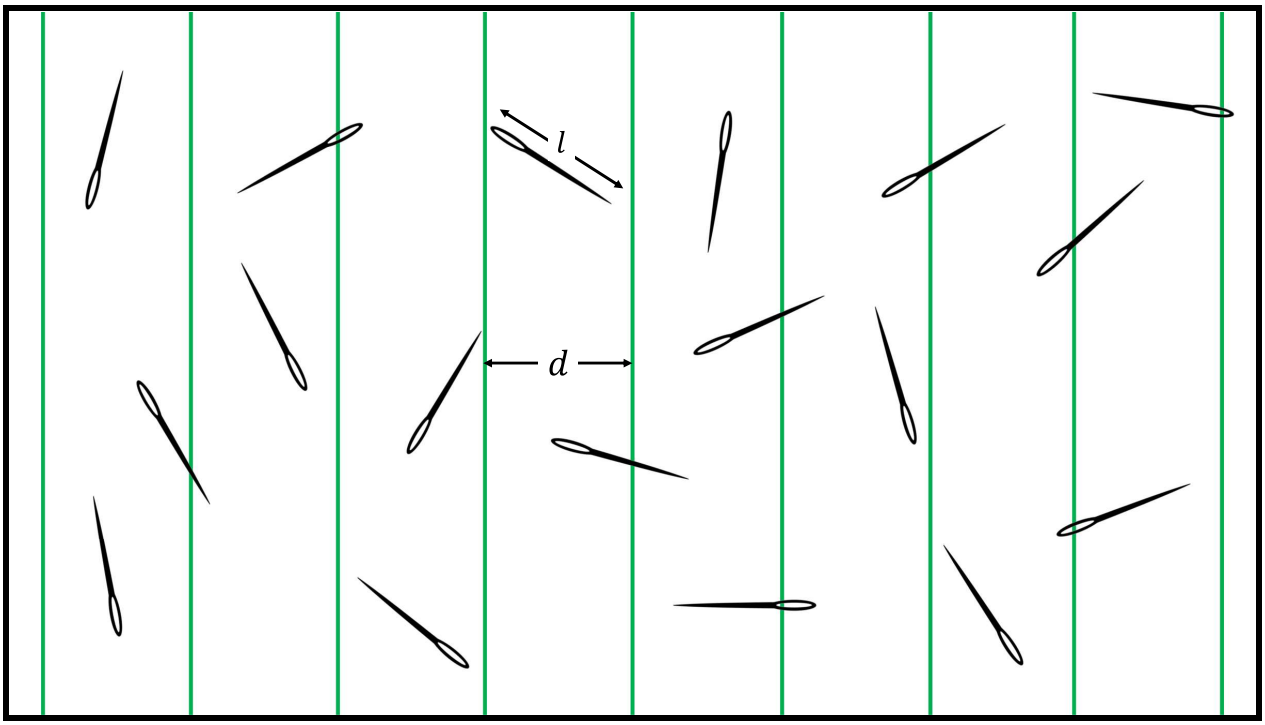


Figure 2.1: Buffon's needle. Consider a set of equally spaced parallel lines on the floor with a distance of d between each line, and needles of length l. Buffon's needle problem involves finding the probability of a randomly thrown needle intersecting one of the lines on the floor. For a short needle, which means $l \leq d$, we can show that this probability is given by $P = \frac{2l}{\pi d}$. To estimate the value of π, we can rearrange this equation as $\pi = \frac{2l}{Pd}$ and estimate the value of P by randomly throwing needles on the floor and tallying the ratio of intersections.

2.1.1 Early History of Monte Carlo Methods

The first instance of a numerical approximation that could be classified as an MC method was done during the early 1900s by Lazzarini [98] to reinterpret Buffon's needle experiment to approximate π up to six decimal places (see Figure 2.1).

The official inception of MC methods was done during World War II at the Los Alamos National Laboratory by Stanislav Ulam and John von Neumann [121] in order to calculate neutron diffusion in a sphere of missile material when working on the first hydrogen bomb. This work was the foundation of what would become the Metropolis–Hastings (MH) Markov chain Monte Carlo (MCMC) method [72]. This

problem was first studied using conventional deterministic mathematical methods but without much success due to the sheer complexity of the problem. To tackle this, Ulam instead proposed to use random experiments and the fast new compute powers offered by the ENIAC supercomputer. The rational proposed by Ulam came out to him while convalescing from an illness and wondering what's the probability that a solitaire card game would come out successfully. After much effort to come up with a combinatorial calculation, he simply observed that we can come up with a decent approximation by trying many games and counting the number of successful games. Following this idea, Ulam shared his idea with von Neumann and the two of them started working on a plan to do actual simulation of the equations in play. Being a top secret project, Nicholas Metropolis proposed to use the codename MC methods in reference to the Monte Carlo casino in Monaco where the Ulam's uncle gambled. The first actual implementation of the MC method on the ENIAC computer was done by Metropolis and Ulam [121] and its success was quickly generalized in other projects at Los Alamos.

Since then, MC methods have become a favorite to handle complex problems in physics, chemistry, engineering and finance due to their simplicity and the increase in computational power of modern computers. For a more general idea of the different applications and specializations that MC methods found over the years, the interested reader should consult the following excellent references [49, 94, 159]. These references also contain other MC methods and probabilistic algorithms that will not be discussed in this thesis such as sequential Monte Carlo (SMC) [37], also called particle filtering, and sampling importance resampling (SIR) [56]. While unrelated to the objectives of this thesis, it is worth nothing that both SMC and SIR found applications in CG [44, 184, 15, 146, 114, 105, 166]. Finally, for a more in-depth historic context on the origin of MC methods from one of its creators, we suggest the paper written by Metropolis [120] himself.

In this thesis, we will focus on three Monte Carlo methods, namely Monte Carlo integration, Markov chain Monte Carlo methods and the Walk-on-Spheres algorithm. We will discuss them in the following sections.

2.1.2 Monte Carlo Integrations

MCI [121] is a family of MC methods used to evaluate integrals using random sampling. In its simplest form, MCI generates uniformly distributed random samples over the domain. MCI has many exciting properties that are not shared with its deterministic counterparts like its ability to handle high-dimensional and complicated integrals, simplicity and that it is easy to parallelize.

2.1.2.1 Variance Reduction

Over the years, many variance reduction methods were invented to make MCI more efficient.

The most acclaimed one is importance sampling (IS) [82]. It is a method created to reduce the variance of MCI by distributing samples proportional to the integrand. It assumes that we can sample from this distribution efficiently (either analytically, through rejection sampling [16], inversion sampling [195], MCMC [122] or the triangle-cut parametrization [73]). This method is what gives power to MCI, especially when the integrand is sparse and high dimensional. For this reason, it was widely adopted in CG and extended many times. Neural IS [133] is a method that leverages a neural network as the importance function for reducing variance in MCI. By leveraging the expressive power of neural networks, Neural IS adapts to complex distributions and provides an alternative to traditional IS techniques.

Multiple importance sampling (MIS) [192] is a variance reduction technique that combines multiple importance functions to improve the estimation process. By optimally blending the contributions of each function using a convex combination of

positive weights, known as the balance heuristic, MIS achieves more accurate and efficient estimations. This approach is particularly useful in scenarios where no single importance function provides an ideal sampling distribution, such as in light transport simulations.

Since its initial creation in light transport, MIS has received many extensions. Optimal MIS [91] generalizes MIS to handle negative weights, making the new heuristic truly optimal. Variance-aware MIS [61] enhances MIS balance heuristic by injecting variance estimates of individual techniques. Continuous MIS [199] is a continuous generalization of MIS that allows for a continuum of importance functions. Marginal MIS [200] is an extension of continuous MIS that provides an unbiased approximation of MIS when one or more of the sampling techniques have an intractable marginal density function.

Control variates (CV) [100] is a variance reduction method that involves finding a function that is highly correlated with the integrand and subtracting it from the integrand. This reduces the variance of the estimator by reducing the contribution of the highly correlated function to the estimate. This variance reduction method found numerous applications in light transport [163, 45]. It was also extended by our community through the work of Müller et al. [134] where they use a neural network as control function.

Antithetic sampling [68] generates pairs of samples that are negatively correlated. It can result in lower variance estimator, although it isn't always guaranteed. It was successfully applied in the context of differentiable light transport by Zeltner et al. [205] and Zhang et al. [206].

Stratified sampling is a sampling strategy that aims to spread samples over the region of integration instead of using naive uniform samples placement. Doing so avoids variance resulting from the clumping of samples and favors a better exploration of the space. The most well-known stratified sampling method is the Latin hypercube sampling method [118]. Stratified sampling is still an active area of research as stratification is still a big problem in rendering [62, 28].

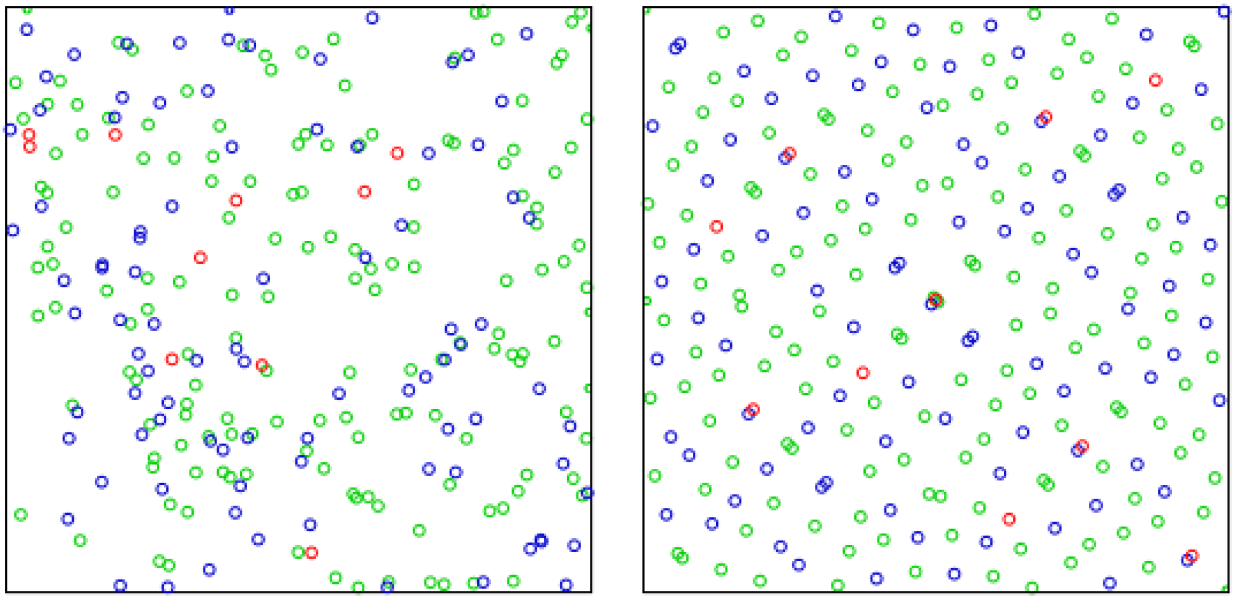


Figure 2.2: Pseudorandom and low discrepancy sequence. Difference between a pseudorandom (left), and a low discrepancy Sobol sequence (right) showing the first 10 samples (red), 100 (red and blue) and 256 (red, blue and green). We observe that the added correlation enables the Sobol sequence to distribute its samples more evenly compared to the pseudorandom one which creates regions with more or less samples - Credit: Jheald, Wikimedia Commons.

The quasi-Monte Carlo (QMC) method uses low-discrepancy sequences instead of pseudorandom (independent) samples, see Figure 2.2. Low-discrepancy sequences are quasi-random or sub-random sequences such that their values are neither random nor pseudorandom but share some properties of random variables. The key idea is to avoid the clumping of samples by deterministically choosing the location of quasi-random numbers and reducing the discrepancy between them. QMC methods use low-discrepancy sequences such as Halton and Sobol sequences. The first adoption of QMC in light transport was due to Keller [90].

In this thesis, we use MCI to devise a novel stochastic estimator of fluid simulations. In doing so, we also provide the first application of IS, CV, antithetic sampling and QMC to fluid animations.

2.1.3 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a class of MC methods used to asymptotically generate samples according to some target distribution using Markov chains. This method is quite useful when we don't have access to a good approximation of the target distribution and when we can't do much more than pointwise evaluation of the integrand. These samples are combined in a similar fashion to MCI for approximate intractable integrals when it's impossible to build a good importance function.

The most used MCMC method is the so-called MH algorithm. It was proposed by Metropolis et al. [122] in the context of statistical mechanics to simulate the Ising model. It's the cornerstone of most of the modern state-of-the-art methods in PBR [193]. Fundamentally, the algorithm generates Markov chains whose states are asymptotically proportional to the target distribution by iteratively sampling a proposal state from a simpler proposal distribution and accepting it based on an acceptance probability that is based on the ratio of the probability of the current and proposed state.

Since its inception, MH has been extended and specialized into many variations of the original algorithm, all with their strengths and weaknesses. We will go over the main MCMC methods but the curious reader can consult Neil's book [139] for a more in-depth discussion. We also illustrate some of these methods in Figure 2.3.

Random walk Metropolis–Hastings (RWMH) simplifies the proposal distribution by assuming it's both isotropic and normally distributed (Gaussian), see Figure 2.3. It's the most widespread and simple implementation of MH. However, RWMH has several limitations. It can be inefficient in high-dimensional problems, or when faced to complex distribution, because the proposal distribution can exhibit low acceptance rates or high autocorrelation between samples. This behavior can result in artifacts and slow convergence. Most of the following extensions of MH focus on solving these problems.

Gibbs sampling [52] is a refinement of MH for high-dimensional distributions that sequentially samples from the conditional distributions of the individual dimensions.

One of the main limitations of Gibbs sampling is that it can be computationally expensive, especially in high-dimensional problems. The algorithm requires the conditional distributions of each variable given the other variables in the model to be known and easily computable. Additionally, it has difficulties to handle dependence between dimensions.

Reversible jump Markov chain Monte Carlo (RJMCMC) [57] is a variant of MH that allows proposals that change the dimensionality of the space. It suffers from the same kind of problems that Gibbs sampling does.

The adaptive Metropolis–Hastings (AMH) [63] algorithm generates a Markov chain by adapting the proposal distribution to the local information of the target distribution. To do so, the proposal distribution is updated adaptively over the course of the simulation based on the information gained from the samples generated so far. One of the challenges in using the AMH algorithm is the choice of the adaptation scheme as a poorly chosen adaptation scheme can break the convergence of the algorithm.

Multiple try Metropolis (MTM) [106] is a method that accelerates MH by allowing for an increase step size and higher acceptance rates. This algorithm generates multiple proposals at each iteration by sampling candidates from different proposal distributions. Then the best candidate is sampled from this set and accepted based on a certain acceptance probability. One of the advantages of MTM is that it can improve the efficiency and reduce the autocorrelation of the chain. However, it comes with the overhead to having to sample multiple candidates at every iteration, which can be detrimental in cases where evaluating the target distribution is expensive.

Metropolis adjusted Langevin algorithm (MALA) [60] is an algorithm that is widely used for sampling from complicated probability distributions. To do so, it generates a Markov chain by combining the MH with a Langevin diffusion process. MALA takes into account the gradient of the target distribution, which can help to guide the proposal distribution towards regions of high probability and to reduce the autocorrelation

between samples. However, a bad choice of the step size parameters can result in overshooting high gradient parts of the distribution. A common solution is to use a truncated version of MALA [6]. In addition, evaluating the gradient of the target isn't always a possibility and can be extremely expensive, even when relying on automatic differentiation. In the latter case, the MALA proposal will not have access to the geometric discontinuities of the target, such as visibility in light transport, which can also hinders its performances.

Manifold Metropolis adjusted Langevin algorithm (MMALA) [54] is a generalization of MALA to Riemannian manifolds, see Figure 2.3. It generates a Markov chain by combining the MH algorithm with a Langevin diffusion process that is constrained to lie on the Riemannian manifold by using the Riemannian metric to generate a proposal distribution that respects the geometry of the manifold. A useful application of this method is to use the inverse target distribution Hessian as a basis for a Gaussian proposal distribution covariance matrix. However, it suffers from the same problems as MALA with the addition of having to evaluate the Hessian of the proposal, which gets more expensive the higher the dimensionality of the target.

Hamiltonian Monte Carlo (HMC) [40] is another technique used for sampling from complex probability distributions, see Figure 2.3. It combines MH with Hamiltonian dynamics to generate a Markov chain by simulating the dynamics of a fictitious particle moving in a high-dimensional space. One of the biggest advantages of this method is its ability to generate efficient proposals that can explore the parameter space much more effectively than other MCMC algorithms. However, this comes at the cost of having to simulate Hamiltonian dynamics, which requires gradient computations and integrating over the whole trajectory, which is quite expensive compared to MALA. Similar to MALA, HMC can also be generalized to Riemannian manifolds [54] to be even more powerful but with the added cost of Hessian computation.

Quasi-Newton Markov chain Monte Carlo (QN-MCMC) [208] is an advanced sampling technique that combines the benefits of quasi-Newton optimization methods and MCMC algorithms, see Figure 2.3. It leverages local curvature information from the quasi-Newton method and gradient computation to adaptively update the proposal distribution according to a window of past samples. When successful this method can rival MMALA with the added geometric information from the chain history. However, this method still requires good samples to provide a good approximation of the local curvature. When this isn't the case, exploration can become worse than more naive counterparts but at a much higher cost.

Kernel adaptive Metropolis–Hastings (KAMH) [171] is a method which embeds Markov chain trajectories into a reproducing kernel Hilbert space (RKHS), allowing the feature space covariance of the samples to inform the choice of the proposal, see Figure 2.3. This method results in a Gaussian proposal whose covariance depends on both the current sample's position and the local statistical properties of the target distribution. This method tries to achieve the same goal as the previous differentiable-based ones, but without computing the gradient and Hessian of the target. However, it's even more sensitive to its parameters and is quite limited when faced to highly anisotropic distributions.

Parallel tempering [182, 115], also called replica exchange, is a method developed to handle sparse multimodal target distributions. Essentially, it runs a base chain with the right target distribution and multiple Markov chains at the same time, each with a different target distribution. In general, the choice of targets for these other chains is an increasingly "hot" version of the original target distribution that is easier to explore. At each step, we exchange the states of each chain based on a MH criterion. The idea is that hotter chains will be easier to explore globally because the separate modes, separate regions of high contributions, overlap together. While this method can drastically increase the mixing of the main chain, it also suffers from the cost of having to run the

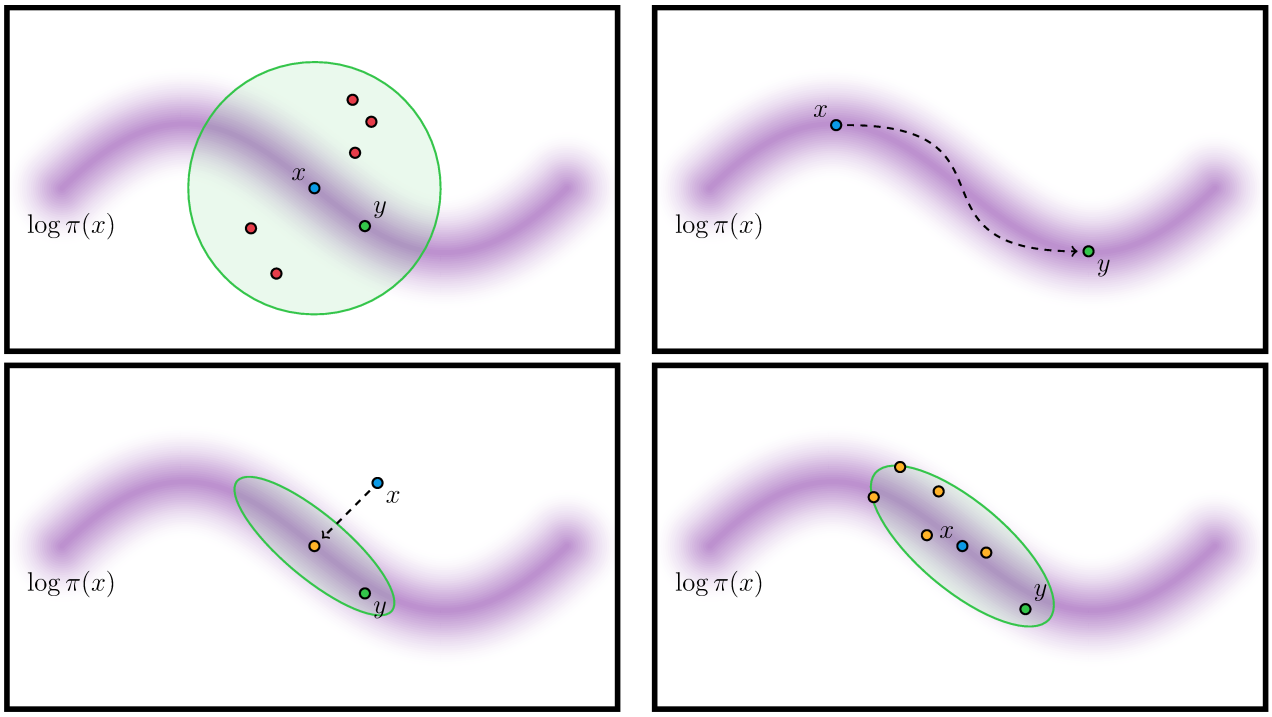


Figure 2.3: Markov chain Monte Carlo methods comparisons. Comparison of different MCMC methods against a complicated probability distribution $\pi(x)$. The current state x of the Markov chain is shown in blue and the next accepted state y in green. Starting with the top left, we have the RWMH approach. This method employs an isotropic proposal distribution (green), which often results in a high volume of proposals that get rejected (shown in red) prior to generating a valid proposal. In the top right, we have the HMC method. This technique leverages Hamiltonian dynamics to craft a trajectory (denoted by the dashed arrow) that leads to a new proposal. This proposal is designed to minimize a predefined Hamiltonian energy (the inverse log-distribution). In the bottom left, the MMALA approach is illustrated. This method harnesses the gradient and Hessian of the log-probability distribution π to construct a shifted anisotropic Gaussian proposal distribution. Finally, in the bottom right, we see the KAMH and QN-MCMC methods. These techniques utilize auxiliary samples (shown in orange) to shape an anisotropic Gaussian proposal distribution.

other chains in parallel. In some cases, however, it is also possible to reweigh the other chains contributions and integrate them to the main estimator [143].

Most interesting to us is the delayed rejection (DR) algorithm. This method improves the MH algorithm by allowing for the use of multiple proposals based on the failure of previous ones in one iteration. It was first introduced by Tierney and Mira [188] and further extended by Mira [124] for fixed dimensional problems and later by Green and Mira [58] for generalized trans-dimensional problems using RJMCMC. Contrary to multiple try Metropolis, this method doesn't need us to sample from every proposal in one step but only one at the time. However, the proposals must be well chosen to avoid problems like the increasingly vanishing acceptance probability over each subsequent stage.

Similar to this method is the delayed acceptance (DA) algorithm [173]. This method first uses a cheap approximation of the target distribution during a first stage as a proxy for the MH acceptance ratio. If the proposal is rejected, we move to the next iteration. If it's accepted, we test a second time with a correction factor accounting for the true target distribution and accept the proposal based on this probability. This method is interesting for problems where the target distribution is expensive to evaluate, such as in light transport. However, the approximation must be really good in order to have gains in efficiency.

In this thesis, we propose an application of the DR algorithm to MLT. This new framework allows us to reduce error while maintaining fast global exploration and low correlation across chains by enabling a two-stage mutation strategy that learns from its mistakes. We will also discuss three unpubilshed projects based on KAMH (Section 6.3.2), QN-MCMC and MMALA (Section 6.3.1), and DA (Section 6.3.3).

2.1.4 Walk-on-Spheres Algorithm

The Walk-on-Spheres (WoS) algorithm, depicted in Figure 2.4, in is a family of MC methods used to approximate the solutions of specific boundary value problems for elliptic partial differential equations. It was first developed by Muller [131] to solve the

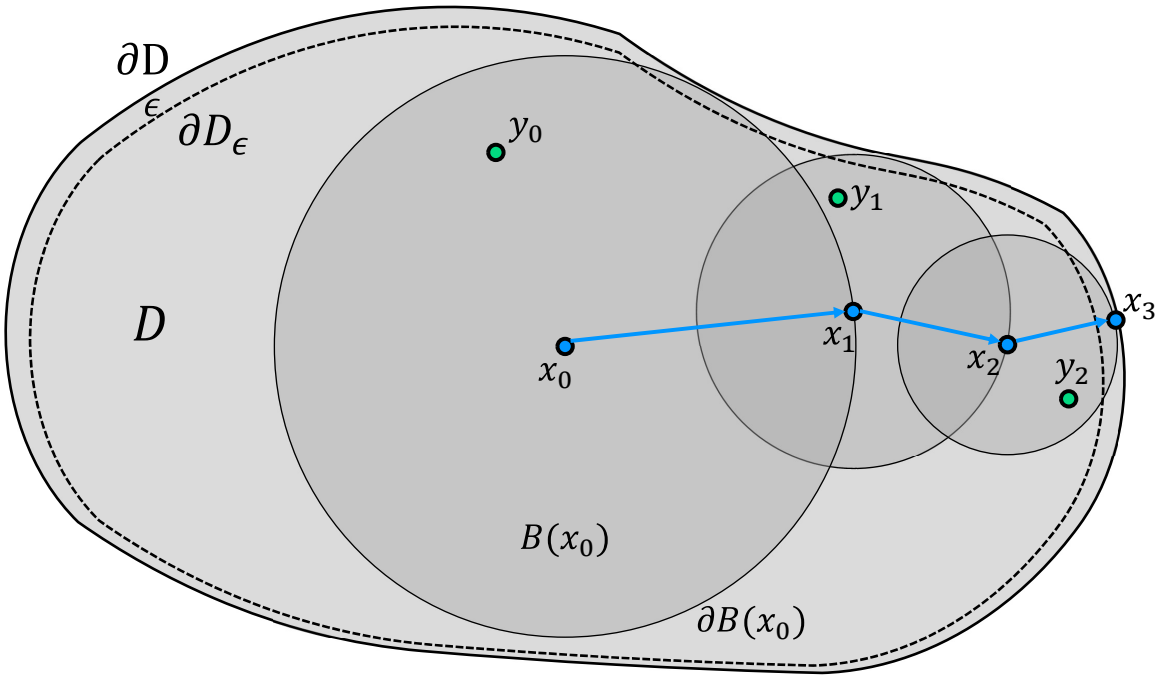


Figure 2.4: Walk-on-Spheres Algorithm. The Walk-on-Spheres algorithm works as follow: First, we start from the position of interest x_0 and find the largest sphere $B(x_0)$ that still fits inside the domain D. We then sample a random point y_0 inside that sphere and add its contribution to a running sum. Then, we sample a new position x_1 on the sphere boundary $\partial B(x_0)$. Once done, we find the largest sphere around this new point and we then iterate this process of adding and sampling until we get within a distance ϵ of the boundary ∂D. Finally, we add the nearest boundary value to the running sum and terminate the recursion.

Laplace equation with Dirichlet boundaries. It relies on probabilistic interpretations of PDEs, and average paths of a diffusion processes in order to compute the pointwise solution of the equation. It has many qualities that usual deterministic approaches don't share like being grid-free, highly parallelizable and robust to the domain complexity.

It was recently introduced to an area of CG known as geometry processing [165]. Similar to the departure of light transport methods from finite element methods (FEM) to MC, they departed from their traditional FEM to WoS. Much like light transport, this had for effect to unlock treatment of geometry that was almost impossible before.

Since this newfound interest, many extensions of the WoS were created. For instance, Sawhney et al. [167] generalized the method to elliptic equations with spatially varying coefficients. Qi et al. [154] used the analogy between forward path tracing and the WoS algorithm to devise a bidirectional method to handle sparse source. Miller et al. [123] created a caching strategy based on the boundary element method to accelerate WoS computations. Finally Sawhney et al. [168] created the Walk-on-Stars algorithm to efficiently handle Neumann boundary conditions.

Inspired by the recent adoption of the WoS algorithm by the geometry community, we apply this framework in the context of fluid simulations in order to handle slip boundary conditions over complex geometry in a grid free manner.

2.2 Physically Based Rendering

In this section, we review the most relevant light transport works to our contributions, namely the MLT algorithm and its extension. For more information on the subject, we refer to Šik and Křivánek [175] which provides a comprehensive survey of MCMC methods in light transport and Pharr et al. [153] for more general information on light transport algorithms and the history of the field. This section is inspired by the related work section that we presented in our previous publication [157].

2.2.1 Early History of Physically Based Rendering

The pioneering application of MCI to light transport revolutionized the field of CG following the formalization of the now foundational rendering equation developed by Kajiya [83]. This led to the development of the basic path tracing algorithm, which represented a major departure from the previously dominant FEM radiosity approaches of the time, such as those proposed by Goral et al. [55]

Over the years, MCI-based path tracing and its variants have continued to evolve, allowing for the treatment of increasingly complex radiometric effects in virtual environments while scaling more gracefully with their growing complexity [153]. Today, modern industrial-caliber renderers rely on path tracing [151] and specialized variance reduction strategies to improve sample efficiency and reduce rendering times.

Notably, the use of path tracing in CG has greatly expanded the scope of photorealistic rendering and has enabled the creation of highly realistic virtual worlds that closely mimic the behavior of light in the physical world. Recent advances in path tracing techniques have also opened up new avenues for real-time rendering in video games [15, 146, 114, 105].

2.2.2 State Spaces

The MLT formulation [193] mutates paths by directly modifying their geometry, such as by removing a path vertex. This approach was simplified by Kelemen et al. [89] with the introduction of primary sample space Metropolis light transport (PSSMLT), which reparameterizes the problem using an abstract sampling domain over the space of random numbers consumed during path generation, see Figure 2.5. In this formulation, paths are represented as vectors of uniform random variables embedded in a unit hypercube, referred to as the primary sample space (PSS).

Concurrent works aiming to unify the two state spaces, path space and PSS, apply inverse mappings from the paths to the random numbers that produced them. For example, Otsu et al. [144] and Pantaleoni [147] convert path space mutation strategies to their PSS representations in a manner that is agnostic to the underlying MCMC framework. In contrast, the reversible jump Metropolis light transport (RJMLT) [14] applies reversible jump methods from statistics [57] to multiplexed Metropolis light transport (MMLT) to map strategies without altering the sample path's geometry. Our

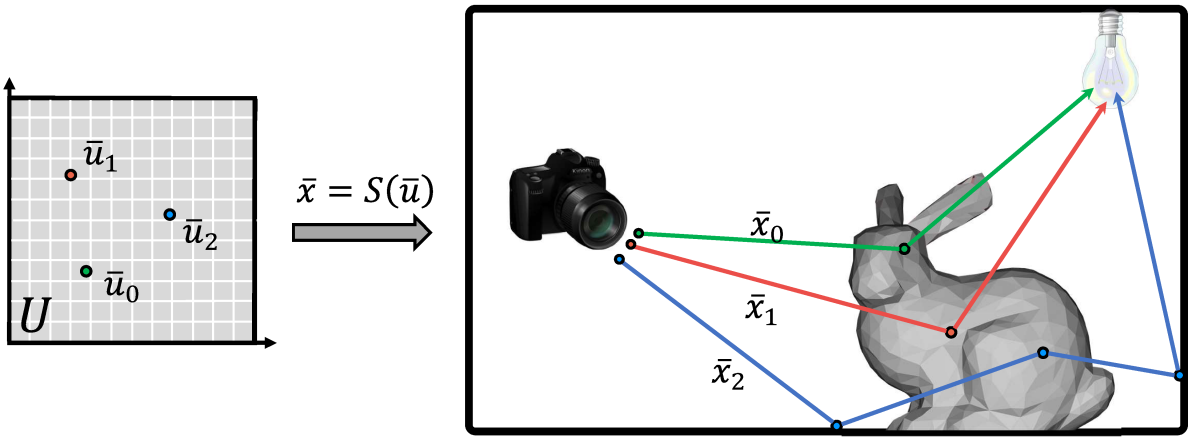


Figure 2.5: Path and primary sample space. In lignt transport a light path $\bar{x}$ is generated via a vector of uniform random numbers $\bar{u}$ and a mapping function S.

method similarly draws from the MCMC literature, adapting and extending DR to MCMC light transport.

When combined with bidirectional path tracing (BDPT) [96, 191], PSSMLT generates a family of paths, many of which may contribute little to no energy. To address this limitation, MMLT [67] augments PSS with an extra dimension indexing across bidirectional sampling techniques so that chains can also explore regions where certain path sampling methods outperform others.

2.2.3 Parallel Tempering

In their work, Otsu et al. [143] apply parallel tempering, also known as replica exchange, to MLT in order to reduce the likelihood of cycling within high energy peaks in the target distribution. At its core, replica exchange tempers many target distributions and mutation strategies to encourage jumps between modes. This requires tracking parallel chains, carefully tuning temperature parameters, and/or judiciously choosing the subspaces over which to spread chains.

In a similar vein, Kaplanyan and Dachsbacher [85] modify the state space by regularizing degenerate densities and delta emission profiles, thereby increasing the likelihood of sampling such narrow interactions.

Building on both of these approaches, Šik and Křivánek [197] extend the regularization idea to improve global exploration while applying parallel tempering to vertex connection and merging.

Our DR method does not rely on parallel tempering and is thus orthogonal to these methods; however, the generality of our algorithm would allow its independent application to any of the tempered chains. It is worth noting that Hachisuka and Jensen's [66] approach, although formulated within the replica exchange framework, can be recast as a special case of DR where the first transition kernel is uniform and the target distribution is the binary visibility.

2.2.4 Specialized Perturbations

Another way of tackling suboptimal exploration is by means of specialized mutations. Li et al. [101] propose a mutation strategy inspired by HMC [40], enabling long traversals in state space and mitigating the chances of getting caught in high energy peaks. Their Hessian Hamiltonian Monte Carlo (H2MC) approach leverages automatic differentiation to perform anisotropic Gaussian mutations, but the added cost of computing gradients and Hessians of the log-target can be significant. Jakob and Marschner [78] model the local differential geometry of path space to design mutations that can explore low-dimensional manifolds of (near) specular chains, with Kaplanyan et al. [86] and Hanika et al. [70] later extending this approach to a more natural half-vector reparameterization. Segovia et al. [170] first suggested multiple-try MCMC [107] to sample paths from several candidates. More recently, Otsu et al. [145] employed mutations that rely on visibility-aware estimates of optimal cone angles at path vertices.

A mutation strategy combining MALA with momentum schemes from stochastic gradient descent procedures was recently developed by Luan et al. [109]. However, it should be noted that our work was published prior to theirs, which is why we didn't include their method in Chapter 4 where we discussed our own approach using Li et al. [101] H2MC method instead. Note that we also worked on a similar unpublished idea (see Section 6.3.1). The goal of this research was to investigate the potential benefits of combining MMALA with a technique known as quasi-Newton MCMC [208], in order to approximate the Hessian matrix of the target density.

While capable of modeling local state space landscapes, specialized transitions all rely on additional technical machinery that tends to increase both the algorithmic complexity of their resulting rendering algorithms and the computational cost per perturbation. Moreover, these mutations are often applicable only to very specific forms of light transport effects. In contrast, our DR method is inexpensive and is capable of effectively performing local exploration only in challenging regions in state space. Most importantly, our method is agnostic to the underlying MCMC perturbation schemes and can be implemented atop any existing MH based algorithms with minimal code modifications.

2.2.5 Adaptive Methods

The AMH algorithm [63] was first introduced to the rendering community by Hachisuka and Jensen [66], where a global mutation size is adapted to light transport complexity during Metropolis-based photon tracing. Zsolnai and Szirmay-Kalos [210] automatically adjust large step probabilities in PSSMLT by gathering acceptance statistics early on during the rendering process. Lai et al. [97] apply population MC to improve sampling based on information collected during initial iterations of an energy-redistribution path tracer [30].

However, these methods are limited to either adjusting algorithm parameters or requiring ad hoc heuristics to maintain ergodicity, as the chains no longer satisfy the Markov property [161]. It is often unclear whether the adjusted parameters will consistently perform well or rather help only on a select set of specialized cases.

In contrast, our two-stage proposal mechanism is similar in spirit to AMH, albeit not strictly adaptive, as it locally changes its behavior on an as-needed basis. Moreover, it maintains detailed balance by construction and does not require any parameter annealing over time. In this regard, our approach is not related to adaptive MCMC but is compatible with such schemes.

While not exactly an adaptive method, Bitterli and Jarosz [13] proposed a technique for selectively using MCMC only in regions that are difficult to explore while using simple MC elsewhere. They do a first pass using a simple transport method, such as MC path tracing, and then use high-variance firefly samples as seeds for MCMC-based techniques, such as MLT. This approach shares a similar goal to the one presented in Chapter 4 of this thesis, which aims to use expensive techniques only where it matters. However, it is worth noting that our work was published before theirs, and so, we did not compare both methods.

2.3 Physically Based Fluid Animation

In this section, we provide an overview of the most significant research in fluid simulation within the field of CG. For a more in-depth exposition, we recommend referring to the comprehensive textbook by Bridson [18], the thesis of Schoentgen [169] and the work of Koschier et al. [92]. The content of this section draws inspiration from the related work section previously presented in our publication [158].

2.3.1 Early History of Physically Based Fluid Animation

The early history of physically based fluid animation can be traced back to several pioneering works in the late 1990s, which laid the foundation for the rapid development of fluid simulation techniques in CG. These works leveraged the Navier–Stokes (NS) equations, the fundamental equations that describe fluid motion, to model various fluid phenomena. The original discovery of the NS equations is attributed to two independent scientists: Claude-Louis Navier and George Gabriel Stokes. Navier published his work in 1822 [138]. Stokes later independently derived and refined the equations in a paper published around 1842-1850 [180]. Both publications contributed to the development of the modern form of the NS equations.

Among the seminal contributions to the field, Stam and Fiume [179] introduced in 1995 the smoothed particles hydrodynamics (SPH) [53, 127] method to the CG community to simulate gaseous phenomena. The original SPH algorithm was initially developed to tackle astrophysical problems. Foster and Metaxas [51] demonstrated in 1996 the simulation of realistic liquid motion, including splashing and pouring behaviors, by employing physically based models. Their paper was one of the earliest examples of using physically based models for fluid animation in CG. In 1999, Stam [178] introduced the stable fluids method, which was a significant breakthrough in the field of fluid animation. Stam's method was based on an unconditionally stable semi-Lagrangian advection scheme that allowed for efficient and visually realistic fluid simulations. His work, published and presented in real-time at the Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH) 1999 conference, greatly influenced the development of fluid animation techniques in CG. In 2001, Fedkiw et al. [46] presented an extension of stable fluids that included vorticity confinement and Catmull-Rom interpolation as a solution to the semi-Lagrangian dissipation from which the previous stable fluids method suffered.

These early works laid the foundation for the rapid development of physically based fluid animation techniques. Over the years, researchers have continued to refine and extend these methods to simulate various fluids phenomena, including water, smoke, fire, and more complex multiphase flows, as well as to develop efficient and robust numerical schemes for solving the underlying equations.

2.3.2 Families of Discretization Methods

The numerical prediction of fluid motions plays an important role in a variety of fields, from weather forecasting to the aerospace industry. In CG, there is significant demand to animate flowing water, swirling smoke, flickering flame, and so on. Most work on this topic falls into a few broad families of methods based on discretizing different forms of the incompressible NS and Euler equations. We illustrate the Eulerian and Lagrangian families in Figure 2.6.

Eulerian methods are a class of numerical techniques used to simulate fluid flow and other dynamic systems by focusing on fixed points in space. They describe the evolution of fluid properties such as velocity, pressure, and density at these fixed spatial locations as the fluid flows through them. In its simplest form, Eulerian methods assume a static mesh or grid through which the fluid flows and the relevant fluid equations are discretized with finite difference/volume/element ideas [50, 178, 47]. This family also extends to method using adaptive grids and meshes but the key concept of fixed points stay the same. They can also be extended to not just keep a value at the cells center but also on edges/faces/corners using staggered marker-and-cell (MAC) grids [71]. This is particularly useful to store vector fields such that they exhibit better conservation properties.

Lagrangian methods are a class of numerical techniques used to simulate fluid flow and other dynamic systems by focusing on individual fluid particles or elements as they move through space. They describe the evolution of fluid properties such as velocity,

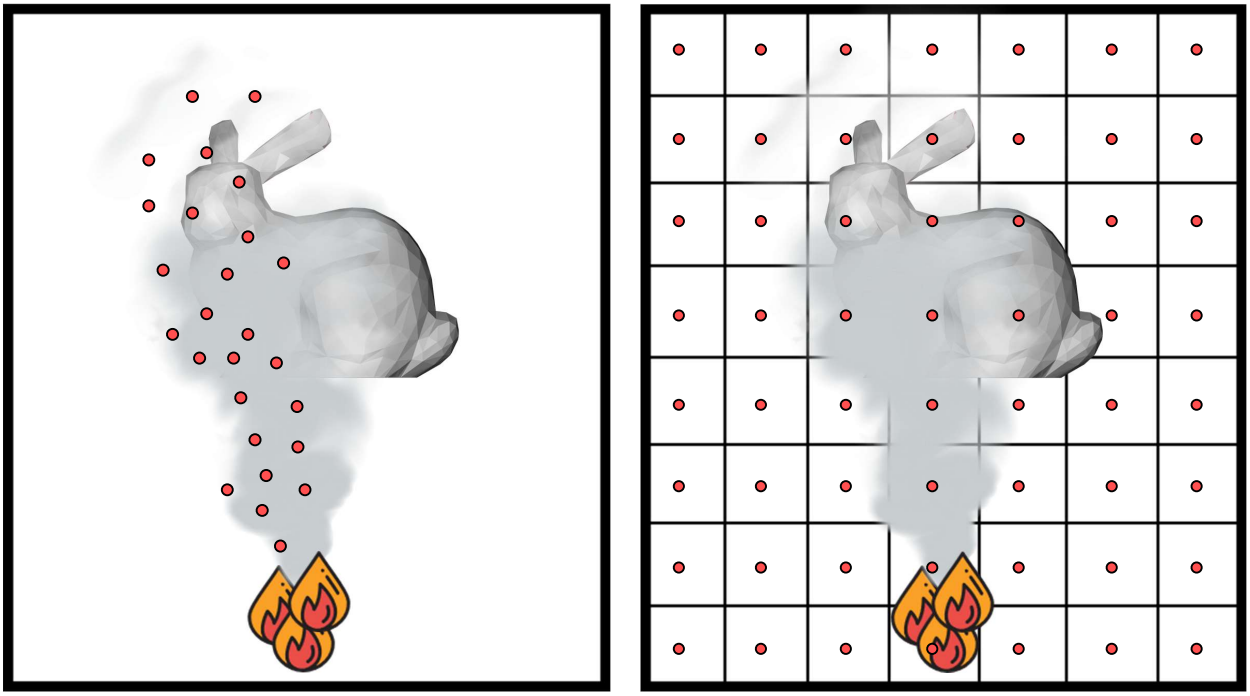


Figure 2.6: Eulerian and Lagrangian methods. Here, we are cooking the Stanford bunny using two distinct types of discretization. On the left, a Lagrangian method is used, which employs a particle representation that moves synchronously with the flow, thereby capturing the dynamic nature of the system. Conversely, on the right, we illustrate an Eulerian method, characterized by the utilization of a fixed uniform grid. This method provides a fixed representation, focusing on changes in system properties at fixed locations over time.

pressure, and density by tracking the motion of these particles or elements over time. Lagrangian methods instead use degrees of freedom that move along with the fluid, often using mesh free particles such as in SPH [53, 127, 179, 39, 129, 177, 76, 10], position based fluid (PBF) [130, 112] and vortex particles [172]. They also encompass time-evolving meshes that must be adapted when they become too deformed [126, 29].

Hybrid methods, which use both Lagrangian particles and Eulerian grids, have become popular because they can offer both the convenience and efficiency of Eulerian grids and the accurate advection of Lagrangian particles like particle-in-cell (PIC) [71], fluid-implicit-particle (FLIP) [17], PIC/FLIP [209], affine particle-in-cell (APIC) [81], and

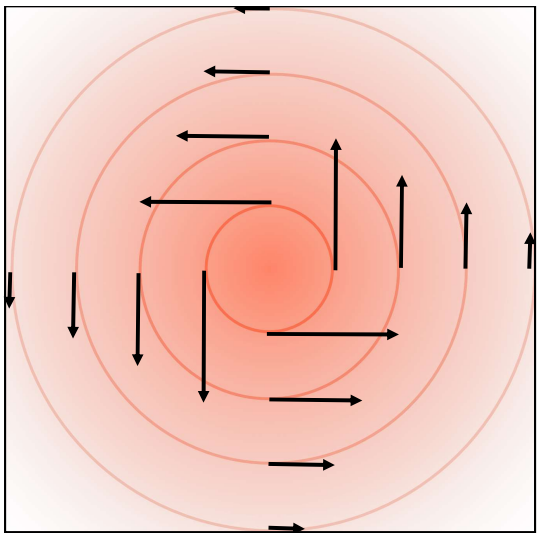


Figure 2.7: Vorticity and Velocity in 2D. Given a velocity vector field $\vec{v}$, shown with the arrows, we define the vorticity field as the scalar field $\omega = \nabla \times \vec{v}$, whose level sets are shown in red.

material point method (MPM) [181, 185, 74], to name a few. The PIC method simulates everything in a grid except the advection step which relies on particles and interpolation but suffers from numerical diffusion. The FLIP method does it the other way around by considering the particles as the main representation but suffers from stability issues. PIC/FLIP does a bit of both by giving the user a choice on the level of viscosity. APIC augments each particle in PIC with a locally affine, rather than locally constant, description of the velocity. MPM was initially proposed as an extension of FLIP with particles carrying material properties that are then solved in the grid.

We will rely, in part, on tracing flow characteristics backwards in time for potentially long periods, reminiscent of a family of recent Eulerian methods [65, 164, 155] that use characteristic mapping [187, 119]. In contrast to ours, these methods use an Eulerian grid and advect velocity rather than vorticity.

Another important family of methods replaces the primitive variables (pressure, velocity) with other variables with useful mathematical properties. Vorticity-based methods (see Figure 2.7) are a common example and can likewise be treated in either an

Eulerian or Lagrangian fashion. In the Eulerian setting, Mullen et al. [128] presented a circulation-preserving scheme viewed through the lens of discrete differential geometry. Stream functions (2D) or vector potentials (3D) are closely related and defined such that velocity is the curl of a potential, and hence inherently incompressible. Bridson et al. [19] used stream functions to design procedural flows. Ando et al. [1] used vector potentials to animate liquids with incompressible bubbles. More broadly, stream function-vorticity methods have a long history in computational fluid dynamics [23, 149] and recent developments seek Eulerian, non-primitive approaches using Clebsch variables [25, 203].

Our method tracks vorticity as the main primitive variables but departs from the traditional Eulerian, Lagrangian and hybrid deterministic methods in favor of MC methods.

2.3.3 Vortex Particle Methods

A Lagrangian treatment of vorticity variables leads to the vortex particle method. This approach was first introduced by Chorin [26]. The particles represent moving point sources of vorticity, and the velocities needed to move the particles are reconstructed via the Biot–Savart (BS) law. The textbook by Cottet and Koumoutsakos [34] provides a thorough introduction. Early adaptations of such ideas to CG use both vortex particles [148] and vortex filaments [3]. Subsequent developments have considered vortex sheets [150, 20] and vortex segment clouds [201]. The computational complexity of naive vortex methods is $O(N^2)$, where N is the number of vortex elements. Several attempts have been made to improve their computational efficiency, such as the vortex-in-cell method [35], fast multipole method [59], and PPPM [207]. More recently, Nabizadeh et al. [137] proposed a new Eulerian velocity-based fluid solver derived from a reformulated Euler equation using covectors. Contrary to most Eulerian approaches, their resulting solver emulates a vortex method without the expensive conversion

between vortical variables and velocities. However, it is worth noting that this publication came out after ours [158].

Our work also exploits the vorticity form of the fluid equations and the BS law to derive a new recursive integral formulation that is amenable to stochastic MC estimation. Doing so enables a wide variety of MC acceleration and/or variance reduction methods, among which we present IS and CVs.

2.3.4 Probabilistic Fluid Models

There has been some research into probabilistic treatments of the NS equations: the Fourier transformation method [79], the Lagrangian flow method [32] and the forward-backward stochastic differential systems (FBSDS) method [22, 38], to name a few. For a more in-depth review, we refer readers to the work of Cruzeiro [36]. However, this body of work primarily aims at the mathematical study of the NS equations, including questions of existence and uniqueness; no practical numerical simulation has been conducted to the best of our knowledge.

We partially use this body of work—most directly that of Busnello et al. [22]—to construct a stochastic, numerical approach to the NS equations that properly treats viscosity and vortex stretching using the Feynman–Kac (FK) formula. In CG, Sawhney et al. [167] used this formula as a basis to transform time-independent varying coefficients of partial differential equation (PDE) to constant coefficients. We instead use it to handle the time-dependent NS equations.

In a quite different vein, there has been a recent resurgence of effort in CG on lattice Boltzmann methods (LBM) [103, 104], including complex obstacle support [111]. These methods employ probability density functions to model the evolution of fluid particles as a kind of deterministic cellular automaton, and are highly parallelizable. While both being based on a probabilistic interpretation of the governing equations of fluid motions, our method isn't related to LBM as this method obeys deterministic rules.

Chapter 3

Background

In this section, we will provide the necessary background, notation, and terminology for the methods presented in this thesis.

First, in Section 3.1 we will cover the basics of Monte Carlo (MC) methods, which will be crucial to understand the subsequent chapters. This will involve a review of probability theory, stochastic processes, Monte Carlo integration (MCI), Markov chain Monte Carlo (MCMC) methods, and the Walk-on-Spheres (WoS) algorithm. We believe that this review is essential, as these complex topics form the foundation of our thesis contributions. Notably, one of our applications involves implementing an advanced MCMC model in the realm of physically based rendering. Furthermore, we propose a novel application of MCI and the WoS method in the context of fluid simulations, based on the Feynman–Kac (FK) formula for an Itô process.

Next, Section 3.2 we will discuss the fundamentals of light transport. This section will address the assumptions made in geometric optics, common domains and measures encountered in light transport, radiometry, surface light transport, and the MC methods used to solve light transport problems, such as Metropolis light transport (MLT).

Lastly, Section 3.3 will explore the primary concepts of fluid simulations. This section will cover the incompressible Navier–Stokes (NS) equations, the vorticity transport equation, the Biot–Savart (BS) law, and semi-Lagrangian discretizations.

Additionally, in Appendix A, we provide a review of essential measure theory topics. These include measure spaces, measurable functions, Lebesgue integration, the Radon–Nikodym theorem, and the most significant measures, like the Lebesgue and pushforward measures. Many of the topics presented in both the MC methods and light transport sections rely on these definitions. We believe that refreshing our understanding of these concepts is important for a thorough comprehension of the material of this thesis.

3.1 Probability and Monte Carlo Methods

The aim of this thesis is to investigate and assess the efficacy of various MC methods in computer graphics (CG), encompassing MCI, MCMC methods, and the WoS algorithm. Since these methods are heavily dependent on a solid grasp of probability theory and stochastic processes, we will commence with a concise review of the fundamental concepts and definitions pertinent to these two fields. This will establish a foundation for our subsequent exploration of the central MC methods presented later in this section, as well as their respective applications.

3.1.1 Probability Theory

In this section, we will present an overview of the fundamental concepts of probability theory pertinent to our work.

We will begin by defining a probability space as a mathematical structure that models the outcomes of a random experiment, followed by an introduction to the concept of conditional probability. Subsequently, we will explain the notion of random variable (RV),

encompassing general, discrete, and continuous types, which are functions mapping the outcomes of a probability space to real numbers. In doing so, we will define the concepts of distribution, independence, expectation, and variance. We will then discuss how to handle functions of RV, and finally, we will state the law of large numbers (LLN), which serves as the foundation for MCI.

3.1.1.1 Probability Space

The most fundamental concept in probability is that of a probability space, which is a mathematical construct that gives meaning to random experiments. In essence, every random experiment can be described using a specific probability space. To help with our understanding we will do a case study based on coin flipping. Intuitively, we need three components to define a probability space:

1. A *sample space*, Ω, which is the set of all possible *outcomes* that the result of a random experiment can take. For instance, in the context of flipping exactly one coin, the outcome can be either head H or tail T, meaning that $\Omega = \{\mathrm{H}, \mathrm{T}\}$.

2. An *event space*, $\mathcal{F}$, which is a set of *events* that are themselves sets of outcomes in the sample space. It is a subset of the power set of all outcomes, e.g., $\mathcal{F} \subseteq 2^{\Omega}$. An event is said to have occurred during an experiment when the outcome of the latter is an element of the event. Continuing with our coin-flipping example, we have $\mathcal{F} = \{\{\}, \{\mathrm{H}\}, \{\mathrm{T}\}, \{\mathrm{H}, \mathrm{T}\}\}$. If the result of the coin flip is H, then both the $\{\mathrm{H}\}$ and $\{\mathrm{H}, \mathrm{T}\}$ events occurred. In fact, the event $\Omega = \{\mathrm{H}, \mathrm{T}\}$ always occurs, and $\{\}$ never occurs.

3. Finally, a *probability measure*, $\mathcal{P}$, which assigns to each event in the event space a *probability*, which is a number between 0 and 1 that describes the likelihood of an event. Intuitively, a probability measure satisfies the properties we expect from the general concept of probabilities: it is normalized, and different outcomes add up

properly when grouped in events. Returning to the coin-flipping example, if the coin is fair, then the probability measure would be $\mathcal{P}(\{\}) = 0$, $\mathcal{P}(\{\mathrm{H}\}) = 0.5$, $\mathcal{P}(\{\mathrm{T}\}) = 0.5$, and $\mathcal{P}(\{\mathrm{H}, \mathrm{T}\}) = 1$. It could also be biased such that $\mathcal{P}(\{\}) = 0$, $\mathcal{P}(\{\mathrm{H}\}) = p$, $\mathcal{P}(\{\mathrm{T}\}) = 1 - p$, and $\mathcal{P}(\{\mathrm{H}, \mathrm{T}\}) = 1$, where p is the frequency at which we obtain H by flipping the coin multiple times in a row and keeping a tally.

More concretely, a ***probability space*** is a triplet $(\Omega, \mathcal{F}, \mathcal{P})$ consisting of:

1. A ***sample space*** Ω that is an arbitrary non-empty set of possible ***outcomes***.
2. An ***event space*** $\mathcal{F}$ that is a sigma-algebra over the sample space Ω whose elements are possible ***events***.
3. A ***probability measure*** $\mathcal{P} : \mathcal{F} \to [0, 1]$ that is a measure that satisfies $\mathcal{P}(\Omega) = 1$.

Starting from an existing probability space, we can also define a new probability measure conditioned on a specific event or restricted to a subset of the sample space. If $(\Omega, \mathcal{F}, \mathcal{P})$ is a probability space and $A \in \mathcal{F}$ is an event with $\mathcal{P}(A) > 0$, we can define a *conditional probability measure* $\mathcal{P}(\cdot \mid A)$ on the restricted event space $\mathcal{F}_A$ by setting

$$\mathcal{P}(B \mid A) = \frac{\mathcal{P}(B \cap A)}{\mathcal{P}(A)}\,, \tag{3.1}$$

for all events $B \in \mathcal{F}_A$. This conditional probability measure describes the probabilities of events given that event A has occurred. Furthermore, if $B \in \mathcal{F}$ is another event with $\mathcal{P}(B) > 0$, we have the following relation, called *Bayes' Theorem*:

$$\mathcal{P}(A)\mathcal{P}(B \mid A) = \mathcal{P}(B \cap A) = \mathcal{P}(B)\,\mathcal{P}(A \mid B)\,. \tag{3.2}$$

3.1.1.2 Random Variable

Now that we have a solid understanding of probability spaces, we are ready to discuss the topic of RVs, which is fundamental to studying random processes. Informally, an RV

is a mathematical function that maps outcomes of a random process to numerical values. To provide more insight, let us revisit the coin flip experiment. Suppose we want to bet money on the outcome of an unfair coin flip, winning 2 Canadian dollars if the result is heads and losing 1 otherwise. We can model this with a function X, called an *RV*, over the sample space $\{\mathrm{H}, \mathrm{T}\}$, with values $X(\mathrm{H}) = 2$ and $X(\mathrm{T}) = -1$. This function can, in turn, define a new probability *distribution* P_X, which answers the question, *"How likely is it that I have x Canadian dollars after one throw?"*. In this example, the distribution is $P_X(2) = \mathcal{P}(\mathrm{H}) = p$, $P_X(-1) = \mathcal{P}(\mathrm{T}) = 1 - p$, and $P_X(x) = 0$ otherwise. We could also ask other questions, like *"How likely is it that I have at least x Canadian dollars after one throw?"*, which would create another probability distribution and RV. In this specific case, the distribution is called the *cumulative distribution function (CDF)* of the RV and is given by

$$F_X(x) = \begin{cases} 0 & \text{if } x < -1\,, \\ p - 1 & \text{if } -1 \leq x < 2\,, \\ 1 & \text{if } x \geq 2\,. \end{cases}$$

We can also ask the question, *"What would be the average value $\mathbb{E}[X]$ that I should expect to obtain after one throw?"*. In this case, it would be $\mathbb{E}[X] = X(\mathrm{H})\mathcal{P}(\mathrm{H}) + X(\mathrm{T})\mathcal{P}(\mathrm{T}) = 2p - (1 - p) = 3p - 1$. If the coin is fair, the average value would then be 0.5 Canadian dollars. As you can see, RVs allow us to ask many questions about a random experiment, which in turn defines new RVs and probability distributions. Now that we have an idea of what RVs and distributions are, let us define these concepts more rigorously.

Consider a probability space $(\Omega, \mathcal{F}, \mathcal{P})$ and a measurable space $(S, \mathcal{S})$, where $S \subseteq \mathbb{R}$ is called a *state space* and $\mathcal{S}$ is the sigma-algebra resulting from the restriction of the Borel algebra $\mathcal{B}$ to $\mathcal{S}$. A *(scalar) RV* is a measurable function $X : \Omega \to S$ that maps each outcome $\omega \in \Omega$ to a *state* $x \in S$. The *distribution* of an RV X induced on the measurable state space

is defined as the pushforward probability measure $P_X : \mathcal{S} \to [0, 1]$ that satisfies

$$P_X(A) = \mathcal{P}(X^{-1}(A)) := \mathcal{P}(\{\omega \in \Omega \,|\, X(\omega) \in A\}) \,, \tag{3.3}$$

for all $A \in \mathcal{S}$. We also define the ***CDF*** $F_X : \mathbb{R} \to [0, 1]$ as

$$F_X(x) := \mathcal{P}(X < x) := \mathcal{P}(X^{-1}(\{(-\infty, x)\})) = \mathcal{P}(\{\omega \in \Omega \,|\, X(\omega) < x\}) \,, \tag{3.4}$$

for all $x \in \mathbb{R}$. We call a ***realization*** the specific value obtained from applying the RV to an observed outcome of an experiment.

We will now discuss some statistics of RV, that are useful in understanding their properties. The ***expectation***, also known as the ***mean***, of an RV X, denoted by $\mathbb{E}[X]$, is the Lebesgue integral of X with respect to the probability measure $\mathcal{P}$:

$$\mathbb{E}[X] = \int_\Omega X(\omega) \, \mathrm{d}\mathcal{P}(\omega) \,. \tag{3.5}$$

Equivalently, it can be rewritten as

$$\mathbb{E}[X] = \int_S x \, \mathrm{d}P_X(x) \,. \tag{3.6}$$

The ***variance*** of an RV X, denoted by $\mathrm{Var}[X]$, is a measure of the spread of the values of X around its mean. The variance is defined as the expectation of the squared difference between X and its mean:

$$\mathrm{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \int_\Omega (X(\omega) - \mathbb{E}[X])^2 \, \mathrm{d}\mathcal{P}(\omega) \,. \tag{3.7}$$

Similar to the expectation, it can be rewritten as

$$\mathrm{Var}[X] = \int_S (x - \mathbb{E}[X])^2 \, \mathrm{d}P_X(x) \,. \tag{3.8}$$

Finally, the ***standard deviation*** of an RV X, denoted by $\sigma[X]$, is defined as the square root of the variance $\sigma[X] = \sqrt{\text{Var}\,[X]}$.

It is also useful to consider how to combine RVs. Consider two RVs X and Y over the same probability space $(\Omega, \mathcal{F}, \mathcal{P})$ and taking values in the measurable spaces $(S_X, \mathcal{S}_X)$ and $(S_Y, \mathcal{S}_Y)$ respectively. The *joint distribution* $P_{X,Y}$ of X and Y is defined using the pushforward measure of the function $Z : \Omega \to \mathcal{S}_X \times \mathcal{S}_Y$ with respect to $\mathcal{P}$, such that $Z(\omega) = (X(\omega), Y(\omega))$:

$$P_{X,Y}(A) = \mathcal{P}(Z^{-1}(A)) := \mathcal{P}(\{\omega \in \Omega \,|\, Z(\omega) \in A\})\,, \tag{3.9}$$

for all $A \in \mathcal{S}_X \times \mathcal{S}_Y$. We define the *marginal distribution of Z with respect to X* by defining a new probability measure $\mathcal{P}_X(B_X) = P_{X,Y}(B_X \times S_Y)$ for all $B_X \in \mathcal{S}_X$. Similarly, we define the *marginal distribution of Z with respect to Y* using the new probability measure $\mathcal{P}_Y(B_Y) = P_{X,Y}(S_X \times B_Y)$ for all $B_Y \in \mathcal{S}_Y$.

Finally, a common characterization of RVs is the notion of independence. Consider a probability space $(S, \mathcal{S}, \mathcal{P})$ with two sub-sigma-algebras $\mathcal{S}_X$, $\mathcal{S}_Y \subset \mathcal{S}$. Then, $\mathcal{S}_X$ and $\mathcal{S}_Y$ are *independent sigma-algebras* if for all $A_X \in \mathcal{S}_X$ and $A_Y \in \mathcal{S}_Y$ the *events are independent*:

$$\mathcal{P}(A_X \cap A_Y) = \mathcal{P}(A_X)\,\mathcal{P}(A_y)\,. \tag{3.10}$$

Two RVs X and Y are said to be ***independent RVs*** if their corresponding sigma-algebras $\mathcal{S}_X$ and $\mathcal{S}_Y$ are independent. When two RVs are independent, we have the following property for the variance of their sum:

$$\text{Var}\,[X + Y] = \text{Var}\,[X] + \text{Var}\,[Y]\ . \tag{3.11}$$

Note that all of these concepts can be extended to *vector valued RV* $X = (X_1, \ldots, X_d)$ where $X_i : \Omega \to S_i$ are all scalar RVs on the same probability space $(\Omega, \mathcal{F}, \mathcal{P})$ but with

different state spaces $S_i \subseteq \mathbb{R}$. Note that these RV X_i do not necessarily have to be *independent*. In this case, the state space is defined as $S = S_1 \times \cdots \times S_d$ and the sigma-algebra is the restriction of the Borel algebra $\mathcal{B}^d$ to $\mathcal{S}$. All previous discussions generalize to vector valued RVs, with the caveat that the CDF function inequality is done in a component-wise fashion.

We will now discuss two specific cases of RVs that are of interest to us, namely discrete and continuous. We will derive all the quantities that we discussed so far in the context of these two cases, which will simplify their definitions.

Discrete: an RV X is called ***discrete*** if it takes its values in some countable subset $S = \{x_1, x_2, \cdots\} \subset \mathbb{R}$. We define the ***probability mass function (PMF)*** $p_X : S \to [0, 1]$ as

$$p_X(x) = \mathcal{P}(X = x)\,. \tag{3.12}$$

We can get back the distribution from the PMF as follows:

$$P_X(A) = \int_A p_X \,\mathrm{d}\# := \sum_{x_i \in A} p_X(x_i)\,. \tag{3.13}$$

This means that the PMF is the Radon–Nikodym derivative of P_X with respect to the counting measure. The expectation is

$$\mathbb{E}\left[X\right] = \sum_{x_i \in S} x_i \, p_X(x_i)\,, \tag{3.14}$$

and the variance

$$\mathrm{Var}\left[X\right] = \sum_{a_i \in S} \left(x_i - \mathbb{E}\left[X\right]\right)^2 p_X(x_i)\,. \tag{3.15}$$

Given two discrete RV X and Y, we can define the ***joint PMF*** $p_{X,Y}(x,y)$ associated to the discrete vector RV $Z=(X,Y)$ as the non-negative function that satisfies

$$P_{X,Y}(A)=\sum p_{X,Y}(x,y)\,, \tag{3.16}$$

where the sum is over all possible (x,y) such that $Z^{-1}((x,y))\in A$. Similarly, we define the ***marginal PMF*** of $Z=(X,Y)$ with respect to X as

$$p_X(x)=\sum_{y\in S_Y} p_{X,Y}(x,y)\,. \tag{3.17}$$

We say that two discrete RVs are independent if the joint PMF is separable as the product of the marginal PMF:

$$p_{X,Y}(x,y)=p_X(x)\,p_Y(x)\,. \tag{3.18}$$

Finally, we define the ***conditional mass function*** of X given $Y=y$, where $p_Y(y)>0$, as

$$p_{X\,|\,Y}(x\,|\,y)=\frac{p_{X,Y}(x,y)}{p_Y(y)}\,. \tag{3.19}$$

Continuous: an RV X is called ***continuous*** if its distribution is *absolutely continuous* with respect to the Lebesgue measure on the real line. For instance, an RV is continuous if it takes its values in some non-enumerable continuous subset $S\subseteq\mathbb{R}^d$. According to the Radon–Nikodym theorem, a continuous RV admits a *probability density function (PDF)* $f_X:S\to[0,\infty]$ such that

$$F_X(A)=\int_A f_X(\boldsymbol{x})\,\mathrm{d}\lambda(\boldsymbol{x})\,. \tag{3.20}$$

The expectation is then expressible as

$$\mathbb{E}\,[X]=\int_S \boldsymbol{x}\,f_X(\boldsymbol{x})\,\mathrm{d}\lambda(\boldsymbol{x})\,, \tag{3.21}$$

and the variance

$$\operatorname{Var}[X] = \int_S (\boldsymbol{x} - \mathbb{E}[X])^2 \, f_X(\boldsymbol{x}) \, \mathrm{d}\lambda(\boldsymbol{x}) \,. \tag{3.22}$$

Given two continuous RVs X and Y, we can define the ***joint PDF*** $f_{X,Y}(x, y)$ associated to the continuous vector RV $Z = (X, Y)$ as

$$F_{X,Y}(A) = \int_A f_{X,Y}(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\lambda((\boldsymbol{x}, \boldsymbol{y})) \,. \tag{3.23}$$

Similarly, we define the ***marginal PDF*** of (X, Y) with respect to X as

$$f_X(x) = \int_{S_Y} f_{X,Y}(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\lambda(\boldsymbol{y}) \,. \tag{3.24}$$

We say that two continuous RV are independent if the joint PDF is separable as the product of the marginal probability density:

$$f_{X,Y}(\boldsymbol{x}, \boldsymbol{y}) = f_X(\boldsymbol{x}) \, f_Y(\boldsymbol{y}) \,. \tag{3.25}$$

Finally, we define the ***conditional density function*** of X given $Y = \boldsymbol{y}$, where $f_Y(\boldsymbol{y}) > 0$, as

$$f_{X \,|\, Y}(\boldsymbol{x} \,|\, \boldsymbol{y}) = \frac{f_{X,Y}(\boldsymbol{x}, \boldsymbol{y})}{f_Y(\boldsymbol{y})} \,. \tag{3.26}$$

3.1.1.3 Functions of Random Variable

Often, we are interested in functions of RVs. Expectation, variance, and standard deviation are three such examples. In general, a new random variable (RV) Y can be created by applying a measurable function $g : \mathbb{R} \to \mathbb{R}$ to the output of an already existing RV X. The distribution of this new RV is given by the pushforward measure in the usual way:

$$P_Y(A) = \mathcal{P}((g(X))^{-1}(A)) := \mathcal{P}(\{\omega \in \Omega \,|\, g(X(\omega)) \in A\}) \,. \tag{3.27}$$

When considering a function of a continuous RV, and if g is invertible and differentiable, then the resulting PDF of Y is given by

$$f_Y(y) = f_X(\vec{g}^{-1}(y)) \left| \frac{\mathrm{d}}{\mathrm{d}y} \vec{g}^{-1}(y) \right| . \tag{3.28}$$

This generalizes to vector-valued functions $\vec{g} : \mathbb{R}^m \to \mathbb{R}^n$ such that $\boldsymbol{y} = \vec{g}(\boldsymbol{x})$ as

$$f_Y(\boldsymbol{y}) = f_X(g^{-1}(\boldsymbol{y})) \left| \det \left[\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{y}} \vec{g}^{-1}(\boldsymbol{y}) \right] \right| , \tag{3.29}$$

where the differential is now understood as the ***Jacobian derivative*** of the inverse function.

Sometimes, we are interested in taking the expectation of the function of an RV $g(X)$ with respect to the underlying distribution P_X of the RV X. We will denote this as

$$\mathbb{E}_{P_X} [g(X)] = \int_S g(x) \, \mathrm{d}P_X(x) . \tag{3.30}$$

We will drop the distribution notation when it is clear from the context. Of course, we can also extend this concept to the variance and standard deviation.

3.1.1.4 Estimator

An ***estimator*** $\langle I \rangle^N$ of some quantity I is defined as a random variable of the form $\langle I \rangle^N = \langle I \rangle^N (X_1, \ldots, X_N)$ where X_i are a set of N not necessarily independent RVs. In this context, X_i are called ***samples***, N is called the ***sample size***, and realizations of $\langle I \rangle^N$ are called ***estimates*** of the quantity I.

The ***bias*** of an estimator is defined as

$$\mathrm{Bias}(\langle F \rangle^N) = \mathbb{E} \left[\langle F \rangle^N - F \right] . \tag{3.31}$$

When $\mathrm{Bias}(\langle F\rangle^N) = 0$ for all N, we say that the estimator is ***unbiased***. Bias refers to the difference between the expected value of an estimator and the true value of the parameter it is estimating. An unbiased estimator has an expected value equal to the true value of the parameter. In other words, on average, an unbiased estimator produces accurate estimates of the parameter.

An estimator is called ***consistent*** if

$$\mathrm{Prob}\left[\lim_{N\to\infty}\left|\langle F\rangle^N - F\right| = 0\right] = 1\,. \tag{3.32}$$

Note that this convergence notation is called ***almost sure convergence*** and means that we have convergence for *almost all* sequences $\langle F\rangle^N$. Consistency refers to the behavior of an estimator as the sample size increases. A consistent estimator converges to the true value of the parameter as the sample size approaches infinity *almost surely*. This means that as more data become available, the estimator's accuracy improves, and its variability decreases.

Similarly, we define the mean squared error (MSE) as

$$\mathrm{MSE}(\langle I\rangle^N) = \mathbb{E}\left[(\langle I\rangle^N - I)^2\right]\,, \tag{3.33}$$

and the root mean squared error (RMSE) as

$$\mathrm{RMSE}(\langle I\rangle^N) = \sqrt{\mathrm{MSE}(\langle I\rangle^N)}\,. \tag{3.34}$$

The main difference between the MSE and RMSE is that the RMSE returns the error in the original units of the data, making it more interpretable. For this reason, the RMSE is often preferred over the MSE. Note that we have the following relation between the MSE, bias

and variance of the estimator:

$$\begin{aligned} \mathrm{MSE}\left(\langle I\rangle^N\right) &= \mathbb{E}\left[\left(\langle I\rangle^N - I\right)^2\right] \\ &= \mathrm{Bias}(\langle I\rangle^N)^2 + \mathrm{Var}\left[\langle I\rangle^N\right] , \end{aligned} \tag{3.35}$$

which means that for an unbiased solver, the MSE is proportional to the variance of the estimator.

3.1.1.5 Convergence Theorem

Let $X_1, \dots X_n$ be n ***independent and identically distributed (IID)*** copies of an RV X and $f(X)$ a measurable function over this RV. Then the ***strong law of large number*** states that

$$\mathrm{Prob}\left[\lim_{n\to\infty}\left|\frac{1}{n}\sum_{k=1}^{n} f(X_k) - \mathbb{E}\left[f(X)\right]\right| = 0\right] = 1\,, \tag{3.36}$$

under mild regularity conditions on f and X. The convergence of MC estimators is a direct consequence of this theorem.

3.1.2 Stochastic Process

On a high level, a stochastic process is a collection of RVs indexed by a parameter, often representing time or space. Such processes are used to model the evolution of random phenomena over time or space and are critical in probability theory, statistics, and various applications in fields such as physics, finance, and engineering.

More formally, a ***stochastic process***, also known as a ***random process***, is a collection of RVs $X_t \,|\, t \in \mathcal{I}$ over some state space $(S, \mathcal{S})$ where $\mathcal{I}$ is an ***index set***. If $\mathcal{I}$ is *discrete*, then X_t is called a ***discrete-time stochastic process***; if $\mathcal{I}$ is *continuous*, then X_t is called a ***continuous-time stochastic process***. In the following section, we will typically use the notation X_t with a capital letter and a subscript index to denote a stochastic process, assuming that the index set $\mathcal{I}$ is clear from the context.

In the subsequent sections, and essentially throughout the remainder of this thesis, we will assume that our stochastic process is composed of a collection of ***continuous RVs***, i.e., $S \in \mathbb{R}^d$ and continuous, which defines a ***continuous-space stochastic process***. Due to this, we will interchangeably use the words density and distribution, as the Radon–Nikodym theorem enforces the existence of a probability density associated with the distribution of these RVs with respect to the Lebesgue measure. In fact, we will use the same symbol to specify both the density and its distribution.

To begin, we will go over Markov processes, which form the foundation of MCMC methods. We will cover the definition of (time-homogeneous) Markov processes, transition kernels, stationary and limiting distributions, reversibility, the detailed balance condition, as well as ergodicity and the ergodic theorem. These concepts ensure the convergence of MCMC estimators under mild conditions.

Next, we will examine the Wiener process, a continuous-time stochastic process that represents the (scaled) limit of random walks and is closely related to diffusion. The Wiener process is characterized by independent and normally distributed increments with zero mean and variance proportional to the time duration. It is fundamental to define the next stochastic process.

Finally, we will discuss Itô processes, which are a class of stochastic processes that include a deterministic and a stochastic component based on Wiener processes. We will also cover the Itô integral, the Itô stochastic differential equation (SDE), the Euler–Maruyama discretization of the Itô process, and lastly, the FK formula, which serves as the foundation for our MC fluid solver.

3.1.2.1 Markov Process and Markov Chains

This section covers the fundamental principles of Markov chains, which are crucial for understanding MCMC methods. Typically, the term *Markov chain* is used for discrete state spaces, while *Markov process* is used for continuous state spaces. However, we will

use both terms interchangeably for our purposes and always assume continuous state space. Additionally, we will consider only *discrete-time* Markov processes without explicitly mentioning this fact.

A ***Markov process***, denoted by X_n, is a (*discrete-time*) stochastic process that satisfies the ***Markov property***. This property means that the probability of transitioning to the next state depends solely on the current state. In other words, the future evolution of the process is entirely determined by its current state and is independent of its past history:

$$P\left(X_{n+1} \in A \,|\, X_n = \boldsymbol{x}_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1\right) = P\left(X_{n+1} \in A \,|\, X_n = \boldsymbol{x}_n\right) , \tag{3.37}$$

where $A \subset \mathcal{S}$. Additionally, if we assume that the transition probabilities do not depend on time, that is, for all $m, n \in \mathbb{N}$, we have

$$P\left(X_{n+1} \in A \,|\, X_n = x\right) = P\left(X_{m+1} \in A \,|\, X_m = x\right) . \tag{3.38}$$

We call a Markov process that satisfies Equation (3.38) a *time-homogeneous Markov process*, and we will assume that all the Markov processes discussed from this point on are time-homogeneous Markov processes. This assumption allows us to define a ***transition kernel*** $k : S \times \mathcal{S} \to [0, 1]$ as a function with the following properties:

(I) For all $n \in \mathbb{N}$, $k(x, A) = P\left(X_{n+1} \in A \,|\, X_n = x\right)$.

(II) For all $x \in S$, $k(x, \cdot\,)$ is a probability measure on $(S, \mathcal{S})$.

(III) For all $A \in \mathcal{S}$, $k(\,\cdot\,, A)$ is a measurable function on S.

We also define the ***transition probability density*** k, or simply ***transition function***, as the density

$$k(x, A) = \int_A k(x, y)\, \mathrm{d}\lambda(y) . \tag{3.39}$$

Intuitively, a transition probability density represents the likelihood of moving from one state to another. Another valid notation for the transition function is the conditional $k(x \mid y)$ to further emphasize the direction in which the chains move.

Next, a ***stationary distribution*** of the Markov chain X_n is a probability distribution π that satisfies

$$\pi(A) = \int_A k(x, y)\, \pi(y)\, \mathrm{d}\lambda(y)\,, \tag{3.40}$$

meaning that the probability distribution remains unchanged after applying the transition kernel. Stationarity implies that if $X_n \sim \pi$ then $X_{n+s} \sim \pi$ for any $s \in \mathbb{N}$. When this occurs, we say that the chain is ***at stationarity***.

A special case of interest to us is when there is a *unique stationary distribution* that is also the *limiting distribution* of the chain. A ***limiting distribution*** of a Markov chain X_n is a distribution π that also satisfies

$$\pi(y) = \lim_{n\to\infty} \int_S k^n(x, y)\, \pi_0(x)\, \mathrm{d}\lambda(x)\,, \tag{3.41}$$

where π_0 is the initial distribution of the process and we define the ***n-step transition function*** as

$$k^n(x, y) = \int_S k^{n-1}(z, y)\, k(x, z)\, \mathrm{d}\lambda(z)\,, \tag{3.42}$$

which describes the likelihood of transitioning from one state to another in exactly n steps.

Certain conditions on k are required for the limiting distributions to also be stationary distributions. We assume that this is the case, and that the unique stationary distribution exists.

To prove that a distribution π is a stationary distribution, a *sufficient condition* is to show that the Markov chain X_n is ***reversible*** with respect to the distribution π, meaning that it satisfies the ***detailed balance*** condition:

$$\pi(x)\, k(x, y) = \pi(y)\, k(y, x)\,, \tag{3.43}$$

for all $x, y \in S$. In other words, a chain is reversible if the rate at which the chain transitions from state x to state y is the same as the rate at which it transitions from state y to state x when it is in stationary distribution.

Finally, we say that a chain is ***ergodic*** if it satisfies the following properties:

(I) **Time-Homogeneous:** The stochastic process satisfies the Markov property.

(II) **Irreducible:** It is possible to reach any state from any other state in a finite number of steps with positive probability.

(III) **Aperiodic:** It does not exhibit deterministic cycles.

(IV) **Positive recurrence:** Every state in the process has a finite expected return time.

(V) **Invariance:** It possesses a unique stationary distribution.

An ergodic Markov chain satisfies a variant of the law of large numbers (3.36), called the ***ergodic theorem***. Here, instead of working with independent samples it works with a sequence of correlated samples. Suppose that we have an ergodic Markov process X_n and a real-valued function $f(X)$ over this process. Then,

$$\text{Prob}\left[\lim_{n\to\infty}\left|\frac{1}{n}\sum_{k=1}^{n} f(X_k) - \mathbb{E}\left[f(X)\right]\right| = 0\right] = 1\,. \tag{3.44}$$

This means that we have convergence for almost all initial states X_1 under mild regularity conditions on f [159]. This result is central to the MCMC method as it gives a convergence criterion. In general, a proposal state acceptance probability is constructed such that the detailed balance condition is satisfied, thereby ensuring the invariance condition is met. Then, the transition kernel is crafted through an intermediate proposal kernel to enforce all the other ergodicity conditions. This is usually achieved by making the proposal kernel a mixture of a small step mutation that explores the space locally, and a global one that allows the chain to move anywhere with a non-zero probability.

3.1.2.2 Wiener Process

A ***Wiener process*** W_t, also called ***Brownian motion***, is a *continuous-time stochastic process* with the following properties:

(I) $W_0 = 0$.

(II) **The process has independent increments:** for any $0 \leq t_1 < t_2 < \cdots < t_n$, the increment $W_{t_i} - W_{t_{i-1}}$ are independent RVs.

(III) **The process has Gaussian increments:** for any $0 \leq s < t$, the increment $W_t - W_s$ is normally distributed with mean 0 and variance $t-s$, i.e., $W_t - W_s \sim \mathcal{N}(0,\, t-s)$.

(IV) W_t is continuous with respect to t.

Intuitively, a Wiener process can be interpreted as the (scaled) limit of a random walk and is closely related to the concept of diffusion.

3.1.2.3 Itô Process

An ***Itô process*** is a continuous-time stochastic process X_t defined using the following stochastic integral equation:

$$\begin{aligned} X_t &= X_0 + \overbrace{\int_0^t a(X_s, s)\,\mathrm{d}\lambda(s)}^{\text{Drift}} + \overbrace{\int_0^t b(X_s, s)\,\mathrm{d}W_s}^{\text{Diffusion}}\,, \\ X_0 &= x\,, \end{aligned} \tag{3.45}$$

where a and b are known functions with suitable regularity conditions and W_s is a Wiener process.

The first integral is called the ***drift*** term and represents the deterministic part of the process. It is just a usual Lebesgue integral. The second integral is called the ***diffusion*** term and represents the randomness of the process. This integral does not correspond to the usual Riemann or Lebesgue integral and is called an ***Itô / Wiener integral***. It is defined

as follows:

$$\int_0^t b(X_s, s)\,\mathrm{d}W_s = \lim_{n\to\infty} \sum_{[t_{i-1},t_i]\in\pi_n} b(X_{t_{i-1}}, t_{i-1})\left(W_{t_i} - W_{t_{i-1}}\right)\,, \tag{3.46}$$

where π_n is a sequence of partitions of $[0, t]$ with lengths going to 0 in the limit.

An Itô process can also be informally represented as an ***Itô SDE***:

$$\begin{aligned} \mathrm{d}X_t &= \overbrace{a(X_t, t)\,\mathrm{d}t}^{\text{Drift}} + \overbrace{b(X_t, t)\,\mathrm{d}W_t}^{\text{Diffusion}}\,, \\ X_0 &= x\,. \end{aligned} \tag{3.47}$$

On a high level, the interpretation of Equation (3.47) is that during a small-time interval $\mathrm{d}t$ the stochastic process changes its value by an amount that is normally distributed according to $\mathcal{N}\left(a(X_t, t),\, b^2(X_t, t)\,\mathrm{d}t\right)$ and this displacement is independent of the past of the process.

3.1.2.4 Euler–Maruyama Discretization

The ***Euler–Maruyama method*** is a numerical scheme for approximating solutions of Itô SDE like Equation (3.47). The method is based on the *Euler method* for ordinary differential equations and is given by the following discretization:

$$X_{n+1} = X_n + f(X_n, t_n)\,\Delta t + g(X_n, t_n)\,\Delta W_n\,, \tag{3.48}$$

where X_n are fixed-time approximation of $X_{n\Delta t}$ with $t_n = n\Delta t$, Δt being called the time step, and $\Delta W_n \sim \mathcal{N}(0, \Delta t)$ being a Gaussian RV with mean 0 and variance Δt.

Note that this is just one of the many numerical methods available for solving SDEs, and its convergence properties depend on the specific characteristics of the SDE being approximated. In the case where g is constant in space and in time, we can safely use any discretization scheme, such as Runge–Kutta, for the deterministic part and add the

Gaussian noise to the result. However, for the general case, extra care must be taken to handle the random part because of the non-intuitive properties of the Wiener process.

3.1.2.5 Feynman–Kac Formula

The FK formula is a powerful result in the theory of stochastic processes that connects a partial differential equation (PDE) with a SDE. It provides a probabilistic representation for the solution of certain types of parabolic PDEs by relating pointwise solutions to the expected value of a function of a stochastic process that solves an associated Itô SDE.

Consider the following time dependent ***Cauchy (initial value) problem for the parabolic PDE***

$$\begin{aligned} \frac{\partial}{\partial t}u(t,x) &= \left[\frac{1}{2}\sum_{i,j} a_{ij}(t,x)\,\frac{\partial^2}{\partial x_i \partial x_j} + \sum_i b_i(t,x)\,\frac{\partial}{\partial x_i} + c(t,x)\right] u(t,x), \quad 0 < t \leq T,\ x \in \mathbb{R}^d\,, \\ u(0,x) &= u_0(x), \quad x \in \mathbb{R}^d\,, \end{aligned} \tag{3.49}$$

with a_{ij}, b_i, and c being known functions with suitable regularity conditions [142, 87], $u(t,x)$ being the unknown function we want to solve for and $u_0(x)$ is a given *initial condition* also with suitable regularity conditions. We can associate an Itô stochastic differentiable equation to this parabolic PDE (3.49):

$$\begin{aligned} \mathrm{d}X_t &= b(t,X_t)\,\mathrm{d}t + \sigma(t,X_t)\,\mathrm{d}W_t\,, \\ X_0 &= x\,, \end{aligned} \tag{3.50}$$

where X_t is a d-dimensional *Itô stochastic process*, $b(t,X_t) = [b_i(t,x)]$ is called the ***drift term*** and characterizes the *deterministic* component of X_t, $\sigma(t,x)$ is the ***diffusion term*** which characterizes the *random* component of X_t and satisfies $\sigma(t,x)\,\sigma^T(t,x) = [a_{ij}(t,x)]$, and finally W_t is a d-dimensional Wiener process. The ***FK formula*** [88, 142] states that the solution $u(t,x)$ of the Cauchy problem for the parabolic PDE (3.49) can be expressed as the

expected value of a function involving the stochastic process X_t that solves the associated SDE (3.50):

$$\mathbb{E}\left[\exp\left(\int_0^t c(s, X_s)\,\mathrm{d}\lambda(s)\right)\, u_0(X_t)\,\middle|\, X_0 = x\right], \tag{3.51}$$

where $\mathbb{E}$ denotes the expectation taken with respect to the probability law of the stochastic process X_t starting at x (i.e., with respect to the Wiener measure μ_W). The integral inside the expectation is a standard Lebesgue integral with respect to time. In practice we can use MCI to solve this expectation by sampling random walks according to the Euler–Maruyama discretization (3.48) with proper variance and averaging their contributions. We will use this result to express the vorticity transport equation as an MC estimator.

3.1.3 Monte Carlo Integration

MCI is a method of numerical integration that uses random sampling to estimate the value of an integral. The basic idea of MCI is to approximate an integral by taking a large number of random samples from the integration domain and computing the average of the function values at these points (Figure 3.1). The more samples are taken, the more accurate the estimate becomes.

This section will discuss MCI by laying the fundamentals of the technique, presenting some of its properties and discussing some variance reduction techniques that can be leveraged to make the estimator more efficient.

More concretely, in many CG applications, we often need to compute the integral F of a complicated and potentially high-dimensional real-valued function $f(\boldsymbol{x})$ over some domain $S \subset \mathbb{R}^d$:

$$F = \int_S f(\boldsymbol{x})\,\mathrm{d}\lambda(\boldsymbol{x})\,. \tag{3.52}$$

In many cases, we do not have access to a closed-form solution of this integral and we must instead rely on a numerical approximation. MCI is an easy to implement approximation method that is largely resilient to the curse of dimensionality, mainly

because its convergence rate relative to the number of samples remains unaffected by the dimensionality of the problem. Additionally, it exhibits a reduced sensitivity to integrand singularities and adapts effectively to complex integration domains when combined with a good sampling strategy. However, it is not entirely immune to these issues. In scenarios involving extremely high-dimensional spaces or functions with problematic behavior, Monte Carlo integration may experience increased overall variance and encounter inefficiencies in sampling.

Intuitively, the method consists of doing a weighted average of the integrand over these random samples using their corresponding densities as weights. More formally, an N-sample ***MC estimator*** $\langle F \rangle^N$ of the integral F is given by

$$\langle F \rangle_{\text{MC}}^N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\boldsymbol{x}_i)}{p(\boldsymbol{x}_i)} \,, \tag{3.53}$$

where p is a PDF, also called an ***importance function*** in this context, and $\{\boldsymbol{x}_i\}_{i=0}^N$ are N independent realizations, also called ***samples***, of the RV X associated to p. This estimator comes from the observation that

$$\int_S f(\boldsymbol{x}) \, \mathrm{d}\lambda(\boldsymbol{x}) = \int_S \frac{f(\boldsymbol{x})}{p(\boldsymbol{x})} \, p(\boldsymbol{x}) \, \mathrm{d}\lambda(\boldsymbol{x}) = \int_S \frac{f(\boldsymbol{x})}{p(\boldsymbol{x})} \, \mathrm{d}p(\boldsymbol{x}) = \mathbb{E}\left[\frac{f(\boldsymbol{x})}{p(\boldsymbol{x})}\right] \,. \tag{3.54}$$

Choosing a good importance function is key for obtaining a good MC estimator as we will see shortly. In the following, we will assume that $p(\boldsymbol{x}) \neq 0$ whenever $f(\boldsymbol{x}) \neq 0$ and that f is regular enough [159]. While this is not a necessity, it does remove some headaches that would not add value to this overview.

3.1.3.1 Properties

Note that the estimator $\langle F \rangle_{\text{MC}}^N$ itself can be interpreted as an RV that depends on N independent copies $\{X_i\}_{i=0}^N$ of the underlying RV X. As such, we can compute its

expectation and variance. For the expectation, we have

$$\begin{aligned}
\mathbb{E}\left[\langle F\rangle_{\text{MC}}^{N}\right] &= \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[\frac{f(X)}{p(X)}\right] \\
&= \frac{1}{N}\sum_{i=1}^{N}\int_S \frac{f(\boldsymbol{x})}{p(\boldsymbol{x})}\,p(\boldsymbol{x})\,\mathrm{d}\lambda(\boldsymbol{x}) \\
&= F\,.
\end{aligned} \tag{3.55}$$

This means that Equation (3.54) is unbiased, $\text{Bias}(\langle F\rangle) = 0$, and always gives back the correct result on average, even with a finite number of samples. For this reason, we can easily combine different MC estimators together while still expecting the same property. Now, for the variance we have

$$\begin{aligned}
\text{Var}\left[\langle F\rangle_{\text{MC}}^{N}\right] &= \frac{1}{N^2}\text{Var}_p\left[\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\
&= \frac{1}{N^2}\sum_{i=1}^{N}\text{Var}_p\left[\frac{f(X)}{p(X)}\right] \\
&= \frac{1}{N}\text{Var}_p\left[\frac{f(X)}{p(X)}\right]\,,
\end{aligned} \tag{3.56}$$

where we used the fact that the samples are independent (Equation (3.11)) to swap the sum and the individual variances. Considering that the MSE of the estimator can be written as

$$\begin{aligned}
\text{MSE}\left(\langle F\rangle_{\text{MC}}^{N}\right) &= \underbrace{\text{Bias}(\langle F\rangle_{\text{MC}}^{N})^2}_{0} + \text{Var}\left[\langle F\rangle_{\text{MC}}^{N}\right] \\
&= \frac{1}{N}\text{Var}_p\left[\frac{f(X)}{p(X)}\right]\,,
\end{aligned} \tag{3.57}$$

and thus, we have that the RMSE converges to 0 at a rate of $\mathcal{O}(N^{-1/2})$. This shows that the convergence of MC estimators does not depend on the dimensionality of the problem

and only on the number of samples used. To reduce the error by a factor of two, we just need to quadruple our samples.

Finally, it is worth observing that, according to the *LLN* (3.36) the MC estimator (3.53) converges almost surely when $N \to \infty$. This means that in addition to being unbiased, it is also consistent.

3.1.3.2 Variance Reduction

As noted before, the rooted mean squared error of the MC estimator is given by

$$\mathrm{RMSE}\left(\langle F\rangle^N_{\mathrm{MC}}\right) = \sqrt{\mathrm{MSE}\left(\langle F\rangle^N_{\mathrm{MC}}\right)} = \sqrt{\frac{1}{N}\mathrm{Var}_p\left[\frac{f(X)}{p(X)}\right]}\,. \tag{3.58}$$

This means that if we are able to reduce the variance of our estimator, we also directly reduce the overall error at equal sample counts. We will present the four most common methods to do so, namely importance sampling, multiple importance sampling (MIS), control variates (CV) and antithetic sampling.

Importance Sampling: The first thing that we can change to reduce the variance of the estimator is the importance function p. For instance, let us consider what happens if we set $p = f/b$ where $b = \int_S f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ is the (constant) normalizing factor of the integral:

$$\begin{aligned}\mathrm{RMSE}\left(\langle F\rangle^N_{\mathrm{MC}}\right) &= \sqrt{\frac{1}{N}\mathrm{Var}_p\left[b\right]} \\ &= 0\,.\end{aligned} \tag{3.59}$$

As we can see, the estimator has zero variance in that case. However, this method is not practical since computing b, which was our initial goal (3.52), is usually not possible. Nonetheless, choosing $p \propto f$ will generally result in estimators with much lower variance than using a naive one. This is what ***importance sampling*** is about. One example of applying this method is to set the probability density as part of the integrand in cases

where it is a separable product of terms, or simply by using an approximation of the integrand obtained during a precomputation phase, or by adapting it over time. In the latter case, extra care must be taken because the samples will no longer be independent if we include past samples information into the importance function that are also used in the estimator itself.

Multiple Importance Sampling: Suppose you have access to a family of K importance functions $\{p_i\}_{i=0}^{k}$ that each matches well the integrand but in different regions of the integration domain. We will refer to these importance functions as ***strategies***. One thing that you can try is to use a convex combination, also called a ***mixture***, of these different strategies as the effective importance function:

$$p = \sum_{i=1}^{k} \alpha_i \, p_i \, , \tag{3.60}$$

where α_i are the weights of the mixture and $\sum_{i=1}^{K} \alpha_i = 1$. In some cases, it might do a good enough job. In most cases, however, finding the right values for the weights α_i will be a difficult task.

As an alternative, you can instead define a new estimator such that

$$\langle F \rangle_{\text{MIS}}^{N_1,\ldots,N_K} = \sum_{i=1}^{K} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(\boldsymbol{x}_{i,j}) \, \frac{f(\boldsymbol{x}_{i,j})}{p_i(\boldsymbol{x}_{i,j})} \, , \tag{3.61}$$

where w_i is the ***(MIS) weight*** of the i-th strategy, N_i are the number of samples of the i-th strategy and $\boldsymbol{x}_{i,j}$ is the j-th sample of the i-th strategy. This estimator is called a ***multiple importance*** sampling estimator.

The resulting estimator will be unbiased so long as $\sum_{i=1}^{K} w_i = 1$ and $w_i(\boldsymbol{x}) \neq 0$ when $p_i(\boldsymbol{x}) \neq 0$. The most popular families of weights are given by the ***power heuristic***:

$$w_i(\boldsymbol{x}) = \frac{[n_i\, p_i(\boldsymbol{x})]^\beta}{\sum_{j=1}^{K} [n_j\, p_j(\boldsymbol{x})]^\beta}\,, \tag{3.62}$$

where $\beta \geq 0$. When $\beta = 1$ we get the ***balance heuristic***, which is to the most widespread used weight heuristic as it strikes a good balance between optimal variance reduction and maintaining robustness.

Control Variate: The second thing that we can change to lower the estimator variance is the integrand itself. Suppose for instance that you have access to a function g such that you know its integral G. Then, we can rewrite Equation (3.52) as

$$\begin{aligned} F &= \int_S f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \\ &= \int_S g(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \int_S f(\boldsymbol{x}) - g(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \\ &= G + \int_S f(\boldsymbol{x}) - g(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\,. \end{aligned}$$

Using this new formulation of the integration problem, we can create a new MC estimator called a ***CV estimator***:

$$\langle F \rangle_{\mathrm{CV}}^N = G + \langle F - G \rangle_{\mathrm{MC}}^N\,.$$

When g is similar enough to f this estimator will usually have lower variance. Now the question is to decide when to use this method instead of importance sampling. As a rule of thumb, if f/g is nearly constant, use $p = g/G$ as a importance sampling function, and if $f - g$ is nearly constant, use g as a CV. In addition, finding the right importance function for $\langle F - G \rangle$ can even further decrease the variance.

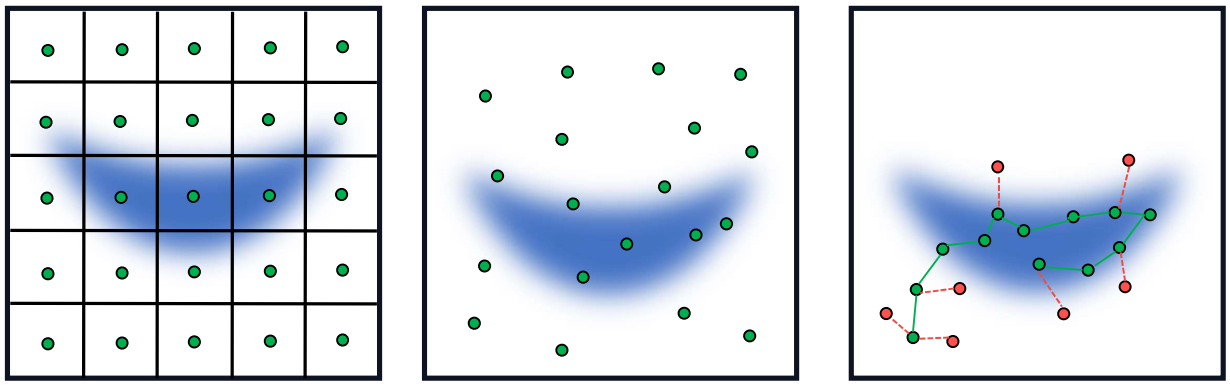


Figure 3.1: Quadrature, Monte Carlo and Markov chain Monte Carlo integration. Usual quadrature integration (left) splits the domain into small deterministic parts,. Monte Carlo integration (middle) samples the integrand at independent random positions. Markov chain Monte Carlo integration (right) samples the integrand by creating a sequence of correlated samples according to some acceptance criterion (green are accepted samples and red are rejected samples).

Antithetic Sampling: In its simplest form, ***Antithetic sampling*** involves averaging two estimators that are *negatively correlated*. For example, if an integrand is very smooth or almost linear, you can sample N pairs of anti-correlated samples $(\boldsymbol{x_i}^+, \boldsymbol{x_i}^-)_{i=0}^N$ and build one estimator $\langle F \rangle_+^N$ with the $(\boldsymbol{x_i}^+)_{i=0}^N$ samples, and another one $\langle F \rangle_-^N$ with $(\boldsymbol{x_i}^-)_{i=0}^N$. Then, you take the average of both estimators:

$$\langle F \rangle_{\text{Anti}}^N = \frac{\langle F \rangle_+^N + \langle F \rangle_-^N}{2} . \tag{3.63}$$

While antithetic sampling can drastically reduce the variance in some contexts, there is no guarantee that the resulting variance will not be worse. Therefore, special care must be taken when using this technique.

3.1.4 Markov Chain Monte Carlo

Again, we want to compute the integral F of a complicated and potentially high-dimensional real-valued function $f(\boldsymbol{x})$ over some domain $S \in \mathbb{R}^d$, e.g.,

$$F = \int_S f(\boldsymbol{x}) \, \mathrm{d}\lambda(\boldsymbol{x}) \,,$$

where λ is the usual Lebesgue measure of $\mathbb{R}^d$ restricted to S.

In Section 3.1.3, we used independent samples to devise an unbiased MC estimator (3.53). However, in some cases, we may not be able to generate independent samples that are proportionally distributed according to the integrand easily. Instead, MCMC methods are designed to generate a series of correlated samples, forming a Markov process, that have the property of being asymptotically distributed according to the normalized integrand, e.g., the target distribution of the Markov chains is set to $\pi = f/b$, where b is the normalizing factor of Equation (3.1.4) (Figure 3.1).

While this approach adds a small amount of bias to the estimator because the samples are no longer independent, it can drastically reduce the amount of noise by generating samples that are distributed proportionally to the integrand without requiring us to sample from the true distribution, achieving a similar effect as doing a perfect importance sampling.

We will present the Metropolis–Hastings (MH) algorithm which is the MCMC method behind the MLT algorithm. This method builds a Markov chain that has the right target distribution by construction. Additionally, this method enables us to only evaluate the integrand without requiring us to evaluate the normalizing factor. Later, we will present an extension of the MLT algorithm that builds upon an extension of MH to a multiple-stage proposal.

3.1.4.1 Metropolis–Hastings

The goal here is to find a method for generating a Markov chain X_n which has a *desired* limiting distribution π that is proportional to the integrant f in Equation (3.1.4). We will refer to this desired distribution π as the ***target distribution***. Many methods exist for constructing chains with a desired limiting distribution. Here we will focus on the ***MH algorithm*** summarized in Algorithm 3.1.

Let the state space be the domain of integration S of our target integral (3.1.4) and set the target density to $\pi = f/b$, where $b = \int_S f(\boldsymbol{x})\,\mathrm{d}\lambda(\boldsymbol{x})$ is the normalizing factor of f over S. We construct a Markov chain by first choosing an initial state $X_1 = \boldsymbol{x}_1 \in S$. Given the n-th realization $\boldsymbol{x}_n$ of the Markov chain X_n, a proposal state $\boldsymbol{y}_n$ is drawn from some ***candidate transition kernel*** $Q(\boldsymbol{x}_n, \cdot\,)$. Here, we will assume the existence of a ***candidate transition density*** $Q(\boldsymbol{y} \,|\, \boldsymbol{x})$ with respect to the Lebesgue measure λ. This step is called the ***proposal step***. Then we accept this proposal state according to the following ***acceptance probability***

$$\alpha(\boldsymbol{x}_n, \boldsymbol{y}_n) = \begin{cases} 1 \wedge \dfrac{\pi(\boldsymbol{y}_n)}{\pi(\boldsymbol{x}_n)} \dfrac{Q(\boldsymbol{x}_n \,|\, \boldsymbol{y}_n)}{Q(\boldsymbol{y}_n \,|\, \boldsymbol{x}_n)} \triangleq r(\boldsymbol{x}_n, \boldsymbol{y}_n), & \text{if } \pi(\boldsymbol{x}_n)\, Q(\boldsymbol{y}_n \,|\, \boldsymbol{x}_n) > 0\,, \\ 1, & \text{if } \pi(\boldsymbol{x}_n)\, Q(\boldsymbol{y}_n \,|\, \boldsymbol{x}_n) = 0\,, \end{cases} \tag{3.64}$$

where $r(\boldsymbol{x}_n, \boldsymbol{y}_n)$ is called the ***acceptance ratio***. To do so, we generate a uniform random number $u \sim U[0, 1]$ between 0 and 1. If $u < \alpha(\boldsymbol{x}_n, \boldsymbol{y}_n)$ the proposed value is accepted and we set $X_{n+1} = \boldsymbol{y}_n$; otherwise, we set $X_{n+1} = \boldsymbol{x}_n$. This step is called the ***acceptance step***. Note that in practice, we don't have to directly evaluate π, and thus b, since the acceptance ratio can be further reduced to

$$r(\boldsymbol{x}_n, \boldsymbol{y}_n) = \frac{f(\boldsymbol{y}_n)}{f(\boldsymbol{x}_n)} \frac{Q(\boldsymbol{x}_n \,|\, \boldsymbol{y}_n)}{Q(\boldsymbol{y}_n \,|\, \boldsymbol{x}_n)}\,, \tag{3.65}$$

as the normalization factor b cancels out.

We can show that this choice of acceptance probability results in a time-homogeneous Markov chain with Markov transition density

$$k(\boldsymbol{x}, \boldsymbol{y}) = \alpha(\boldsymbol{x}, \boldsymbol{y})\, Q(\boldsymbol{y} \,|\, \boldsymbol{x})\,, \tag{3.66}$$

that satisfies the detailed balance condition (3.43), which makes the chain reversible. To do so, we need to consider two cases: if $r(\boldsymbol{x}, \boldsymbol{y}) \geq 1$ then $\alpha(\boldsymbol{x}, \boldsymbol{y}) = 1$ and $\alpha(\boldsymbol{y}, \boldsymbol{x}) = r(\boldsymbol{y}, \boldsymbol{x})$ which gives

$$\begin{aligned}
\pi(\boldsymbol{x})k(\boldsymbol{x}, \boldsymbol{y}) &= \pi(\boldsymbol{x})\, \alpha(\boldsymbol{x}, \boldsymbol{y})\, Q(\boldsymbol{y} \,|\, \boldsymbol{x}) \\
&= \pi(\boldsymbol{x}) \cdot 1 \cdot Q(\boldsymbol{y} \,|\, \boldsymbol{x}) \\
&= \pi(\boldsymbol{x})\, \frac{\pi(\boldsymbol{y})}{\pi(\boldsymbol{y})}\, \frac{Q(\boldsymbol{x} \,|\, \boldsymbol{y})}{Q(\boldsymbol{x} \,|\, \boldsymbol{y})}\, Q(\boldsymbol{y} \,|\, \boldsymbol{x}) \\
&= \pi(\boldsymbol{y})\, \frac{\pi(\boldsymbol{x})}{\pi(\boldsymbol{y})}\, \frac{Q(\boldsymbol{y} \,|\, \boldsymbol{x})}{Q(\boldsymbol{x} \,|\, \boldsymbol{y})}\, Q(\boldsymbol{x} \,|\, \boldsymbol{y}) \\
&= \pi(\boldsymbol{y})\, \alpha(\boldsymbol{y}, \boldsymbol{x})\, Q(\boldsymbol{x} \,|\, \boldsymbol{y}) \\
&= \pi(\boldsymbol{y})\, k(\boldsymbol{y}, \boldsymbol{x})\,.
\end{aligned} \tag{3.67}$$

Similarly, if $r(\boldsymbol{x}, \boldsymbol{y}) < 1$ then $\alpha(\boldsymbol{x}, \boldsymbol{y}) = r(\boldsymbol{x}, \boldsymbol{y})$ and $\alpha(\boldsymbol{y}, \boldsymbol{x}) = 1$ which gives

$$\begin{aligned}
\pi(\boldsymbol{x})k(\boldsymbol{x}, \boldsymbol{y}) &= \pi(\boldsymbol{x})\, \alpha(\boldsymbol{x}, \boldsymbol{y})\, Q(\boldsymbol{y} \,|\, \boldsymbol{x}) \\
&= \pi(\boldsymbol{x}) \frac{\pi(\boldsymbol{y})}{\pi(\boldsymbol{x})}\, \frac{Q(\boldsymbol{x} \,|\, \boldsymbol{y})}{Q(\boldsymbol{y} \,|\, \boldsymbol{x})}\, Q(\boldsymbol{y} \,|\, \boldsymbol{x}) \\
&= \pi(\boldsymbol{y}) \cdot 1 \cdot Q(\boldsymbol{x} \,|\, \boldsymbol{y}) \\
&= \pi(\boldsymbol{y})\, \alpha(\boldsymbol{y}, \boldsymbol{x})\, Q(\boldsymbol{x} \,|\, \boldsymbol{y}) \\
&= \pi(\boldsymbol{y})\, k(\boldsymbol{y}, \boldsymbol{x})\,.
\end{aligned} \tag{3.68}$$

This shows that π is the unique invariant limiting distribution of the chain.

Algorithm 3.1: METROPOLIS $(\boldsymbol{x}_n)$

$\boldsymbol{y}_n \sim Q(\cdot \mid \boldsymbol{x}_n)$
$u \sim U[0,1]$
$\alpha \leftarrow \alpha(\boldsymbol{x}_n, \boldsymbol{y}_n)$ Equation (3.65)
if $u < \alpha$ **then** $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{y}_n$
else $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_n$
return $\boldsymbol{x}_{n+1}$

Finally, as long as Q is chosen so that the chain is ergodic (see Section 3.1.2.1), we have the following estimator

$$\langle F \rangle_{\mathrm{MH}}^{N} = \frac{1}{N} \sum_{k=1}^{N} f(X_k)\,, \tag{3.69}$$

whose convergence is guaranteed by the ergodic theorem (3.44). Note that the convergence rate produced by such a chain depends on both the choice of proposal q and the regularity of the target distribution π [108]. The estimator (3.69) is also slightly biased due to the samples not being independent, although this effect is negligible so long as the chain starts with an initial state that is already distributed according to π. This is usually achieved by using a *burn-in phase* where we discard the first few samples, or during a precomputation phase.

Later, we will extend the application of MH to light transport by instead considering the *delayed rejection* framework as an alternative.

3.1.5 Walk-on-Spheres

The WoS algorithm [131] is an MC method used to estimate solutions of *boundary value problems for linear elliptic differential equations*. In this section, we will present the core of the WoS algorithm, at least as of the date of the acceptance of our publication that is based on it. To do so, we will do an overview of the Poisson equation, Green's and Poisson's functions, the mean-value theorem, the core WoS algorithm and its Gradient variation.

In particular, the WoS algorithm is employed to generate a *pointwise* stochastic approximations of the solution of ***Poisson equation with Dirichlet boundaries*** [42]:

$$\begin{aligned} \nabla^2 \Psi(\boldsymbol{x}) &= f(\boldsymbol{x}) \quad \text{in } \Omega\,, \\ \Psi(\boldsymbol{x}) &= g(\boldsymbol{x}) \quad \text{on } \partial\Omega\,, \end{aligned} \tag{3.70}$$

where ∇^2 is the Laplace operator, Ψ is the unknown function, $\Omega \subset \mathbb{R}^d$ is the domain, $\partial\Omega$ the boundary, f is the ***source function*** and g is the ***prescribed boundary function***. To better understand the method, we will first introduce the ***Laplacian Green's function*** $G^{\Omega}_{\boldsymbol{y}}(\boldsymbol{x})$, which is defined as the solution of the following Laplace equation [43]

$$\begin{aligned} \nabla^2_{\boldsymbol{x}} G^{\Omega}(\boldsymbol{x}, \boldsymbol{y}) &= \delta_{\boldsymbol{y}}(\boldsymbol{x}) \quad \text{in } \Omega\,, \\ G^{\Omega}(\boldsymbol{x}, \boldsymbol{y}) &= 0 \quad \text{on } \partial\Omega\,. \end{aligned} \tag{3.71}$$

We also define the ***Poisson kernel*** $P^{\Omega}(\boldsymbol{x}, \boldsymbol{y})$ as the normal derivative of the Green function over the boundary:

$$P^{\Omega}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\partial}{\partial \hat{n}(\boldsymbol{y})} G^{\Omega}(\boldsymbol{x}, \boldsymbol{y})\,. \tag{3.72}$$

Using these two definitions it is possible to express pointwise solutions [43, Section 2.2.4] of Equation (3.70) as

$$\Psi(\boldsymbol{x}) = \underbrace{\int_{\partial B(\boldsymbol{x})} g(\boldsymbol{y})\, P(\boldsymbol{x}, \boldsymbol{y})\, \mathrm{d}\boldsymbol{y}}_{\text{Boundary}} + \underbrace{\int_{B(\boldsymbol{x})} f(\boldsymbol{y})\, G(\boldsymbol{x}, \boldsymbol{y})\, \mathrm{d}\boldsymbol{y}}_{\text{Volume}}\,, \tag{3.73}$$

where $B(\boldsymbol{x}) \subset \Omega$ is the largest sphere centered at position $\boldsymbol{x}$ and touching the region boundary $\partial\Omega$, see Figure 2.4. We denote the radius of $B(\boldsymbol{x})$ as $R_{\boldsymbol{x}}$ and its boundary as $\partial B(\boldsymbol{x})$. We write the Green's function over the ball, also called the ***harmonic Green's function***, as $\mathcal{H}^{\boldsymbol{x}}\left(\|\boldsymbol{y} - \boldsymbol{x}\|\right) = \mathcal{G}^{B(\boldsymbol{x})}(\boldsymbol{x}, \boldsymbol{y})$. The harmonic Green's function is one of the few Green's functions that is known analytically and its associated Poisson kernel is simply

the inverse area of the boundary $1/|\partial B(\boldsymbol{x})|$. It is possible to express pointwise solution $\Psi(\boldsymbol{x})$ using the harmonic Green's function as the following recursive integral equation (a generalization of the *mean-value theorem*) [43]

$$\Psi(\boldsymbol{x}) = \underbrace{\frac{1}{|\partial B(\boldsymbol{x})|} \int_{\partial B(\boldsymbol{x})} \overbrace{\Psi(\boldsymbol{y})}^{g(\boldsymbol{y}) \text{ if } \boldsymbol{y} \in \partial\Omega} \, \mathrm{d}\boldsymbol{y}}_{\text{Boundary}} + \underbrace{\int_{B(\boldsymbol{x})} f(\boldsymbol{y}) \mathcal{H}^{\boldsymbol{x}} \left(\|\boldsymbol{y} - \boldsymbol{x}\| \right) \, \mathrm{d}\boldsymbol{y}}_{\text{Volume}} . \tag{3.74}$$

The ***WoS*** MC estimator of $\Psi(\vec{x_0})$ [42] is given by

$$\langle \Psi(\boldsymbol{x}_0) \rangle_{\text{WoS}}^{N} = \frac{1}{N} \sum_{i=1}^{N} \widehat{\Psi}(\vec{x_0}) \, , \tag{3.75}$$

where $\widehat{\Psi}(\vec{x_0})$ is a realization of using 1-sample MC estimators in Equation (3.75) and follows the recursion

$$\widehat{\Psi}(\boldsymbol{x}_k) = \begin{cases} g(\boldsymbol{x}_k) & \text{if } \boldsymbol{x}_k \in \partial\Omega \, , \\ \dfrac{\widehat{\Psi}(\boldsymbol{x}_{k+1})}{|\partial B(\boldsymbol{x}_k)| p_{\partial B(\boldsymbol{x}_k)}(\boldsymbol{x}_{k+1})} + \dfrac{f(\boldsymbol{y}_k) \mathcal{H}^{\boldsymbol{x}_k} \left(\|\boldsymbol{y}_k - \boldsymbol{x}_k\| \right)}{p_{B(\boldsymbol{x}_k)}(\boldsymbol{y}_k)} & \text{otherwise} \, , \end{cases} \tag{3.76}$$

with $\boldsymbol{x}_{k+1} \sim p_{\partial B(\boldsymbol{x}_k)}$ sampled from a distribution over $\partial B(\boldsymbol{x}_k)$ and $\boldsymbol{y}_k \sim p_{B(\boldsymbol{x}_k)}$ from a distribution over $B(\boldsymbol{x}_k)$. In practice, we stop the recursion when a sample crosses an ϵ-*shell* inscribed inside the boundary, $\partial\Omega_\epsilon$, or when a threshold on the number of iterations is met. We then project the last point to the nearest boundary location and evaluate the boundary function g there.

Sawhney and Crane [165] showed that it is also possible to directly apply a WoS estimator to compute a pointwise MC approximation of the gradient of the solution $\nabla_{\boldsymbol{x}} \Psi(\boldsymbol{x})$. To do so, they derived the following recursive integral equation using *Malliavin calculus* [9]

$$\nabla_{\boldsymbol{x}} \Psi(\boldsymbol{x}) = \frac{1}{|B(\boldsymbol{x})|} \int_{\partial B(\boldsymbol{x})} \Psi(\boldsymbol{y}) \widehat{n}(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} + \int_{B(\boldsymbol{x})} f(\boldsymbol{y}) \nabla_{\boldsymbol{x}} \mathcal{H}^{\boldsymbol{x}} \left(|\boldsymbol{y} - \boldsymbol{x}| \right) \, \mathrm{d}\boldsymbol{y} \, . \tag{3.77}$$

The ***gradient WoS*** MC estimator is then given by a construction similar to Equation (3.76):

$$\langle \nabla_{\boldsymbol{x}} \Psi(\boldsymbol{x}_0) \rangle_{\text{WoS}}^{N} = \frac{1}{N} \sum_{i=1}^{N} \widehat{\nabla_{\boldsymbol{x}} \Psi}(\vec{x_0}) \,, \tag{3.78}$$

where $\widehat{\nabla_{\boldsymbol{x}} \Psi}(\vec{x_0})$ is a realization of using 1-sample MC estimators in Equation (3.75) and follows the recursion

$$\widehat{\nabla_{\boldsymbol{x}} \Psi}(\boldsymbol{x}_0) = \frac{\widehat{\Psi}(\boldsymbol{x}_1)\hat{n}(\boldsymbol{x}_1)}{|B(\boldsymbol{x}_0)| p_{\partial B(\boldsymbol{x}_0)}(\boldsymbol{x}_1)} + \frac{f(\boldsymbol{y}_0) \nabla_{\boldsymbol{x}_0} \mathcal{H}^{\boldsymbol{x}_0}\left(\|\boldsymbol{y}_0 - \boldsymbol{x}_0\|\right)}{p_{B(\boldsymbol{x}_k)}(\boldsymbol{y}_k)} \,, \tag{3.79}$$

with $\boldsymbol{x}_1 \sim p_{\partial B(\boldsymbol{x}_0)}$, $\boldsymbol{y}_0 \sim p_{B(\boldsymbol{x}_0)}$, and $\hat{n}(\boldsymbol{x}_1) = \widehat{\boldsymbol{x}_1 - \boldsymbol{x}_0}$ is the unit normal of $\partial B(\boldsymbol{x}_0)$ at $\boldsymbol{x}_1$.

Later, we will use Equation (3.75) and Equation (3.78) to define a stochastic estimator to handle slip boundary conditions when solving the NS equations using our MC estimator.

3.2 Fundamentals of Light Transport

Light transport, also known as rendering or image synthesis, is a field of physics that describes the rules governing the propagation and transformation of light as it interacts with matter and is eventually detected by sensors, such as our eyes or a camera film. These interactions can involve emission (when light is produced), reflection (when light bounces off surfaces), refraction (when light passes through translucent materials and changes direction), absorption (when light is absorbed by a medium), and transmission (when light passes through a medium).

Fundamentally, light transport is the process of converting a three-dimensional (3D) scene into a two-dimensional (2D) image composed of a rectangular grid of pixel values. A scene is essentially a data structure containing information about the camera's position, orientation, and settings; light sources, including their intensity, color, and

distribution; and the geometry and materials of objects as well as participating media, which determine how they interact with light.

In this section, we will present the basic concepts of light transport that will be used later in this thesis. This includes a brief overview of the most commonly used domains and measures considered in radiometry and light transport, important radiometric quantities, surface light transport, and MC methods in light transport. This section serves as a summary of several chapters from Veach [190].

3.2.1 Assumptions

In CG, it is common to assume that light behaves according to the principles of *geometric optics*. This model represents light propagation using *rays*, which depicts the shortest path a photon would take in a medium, without accounting for wave-like effects such as diffraction and interference. In this thesis, we will assume that light adheres to the laws of geometric optics.

Furthermore, we will introduce several simplifying assumptions, although it would be possible to account for more complex effects. These assumptions do not affect the methods developed in this thesis and significantly streamline the notation and exposition in this section. First, we will not consider effects such as polarization, fluorescence, and phosphorescence. This means that we will assume that light behaves independently at a specific wavelength and time. Second, we will work in RGB color space instead of the full-wavelength spectrum, which implies that emitted photons can only be in the red, green, or blue wavelength. Lastly, we will focus on surface-only light transport, assuming that empty space consists of a vacuum.

3.2.2 Domain and Measures

To study light transport, it is essential to have a solid understanding of the different integration domains encountered in light transport and their corresponding measures. Radiometry primarily considers two main spaces: the scene space $\mathcal{M}$ and the direction space $\mathbb{S}^2$.

First, let us assume that the scene geometry consists of a finite set of k surfaces $M_i \subset \mathbb{R}^3$. We define their union as the ***scene space*** $\mathcal{M} \subset \mathbb{R}^3$:

$$\mathcal{M} = \bigcup_{i=1}^{k} M_i \,. \tag{3.80}$$

This union includes the geometry of the sensor, emitters, and objects in the scene. In practice, the surfaces must be piecewise differentiable two-dimensional manifolds, including their boundaries to prevent gaps between adjacent surface pieces. The manifold assumption also allows us to define an ***area measure*** A on $\mathcal{M}$ as

$$A(S) = \sum_{i=1}^{k} A_i(S \cap M_i) \,. \tag{3.81}$$

Here, $S \subset \mathcal{M}$ and each A_i denotes the Lebesgue area measure over the specific piece, obtainable from the pushforward measure of $\mathbb{R}^2$ onto the manifolds M_i.

Secondly, ***directions*** are represented as unit-length vectors $\vec{\omega} \in \mathbb{R}^3$ residing on the ***unit sphere***, i.e., $\vec{\omega} \in \mathbb{S}^2$. We denote by Ω the natural area measure defined on $\mathbb{S}^2$, also called the ***solid angle measure*** (see Figure 3.2 and Figure 3.3). This measure can also be derived by the pushforward measure as the unit sphere $\mathbb{S}^2$ is also a manifold. In general, we will consider the solid angle subtended by a surface $S \subset \mathcal{M}$ viewed from a point $\boldsymbol{x}$. This is determined by projecting S onto the unit sphere $\mathbb{S}^2(\boldsymbol{x})$ centered at $\boldsymbol{x}$, and computing the measure of the resulting set of directions. For this reason, we will often write $\Omega_{\boldsymbol{x}}$ to emphasize this view dependence.

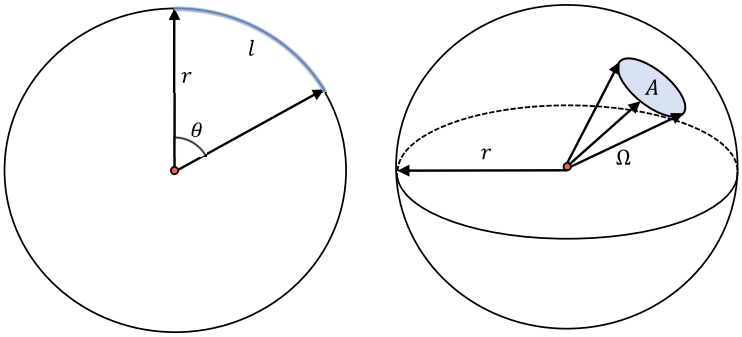


Figure 3.2: Geometric relation between solid angle and surface. The solid angle is a generalization of the usual concept of a 2D angle to 3D. Left: Consider a circle of radius r and an angle θ and its subtend arc length l, then we have that $\theta = \frac{l}{r}$. Right: Similarly, consider a sphere of radius r and a section of area A that subtend the solid angle Ω, then we have $\Omega = \frac{A}{r^2}$.

In addition to these two measures, we will also consider some derived measures called the projected solid angle measure and the projected area measure. First, the ***projected solid angle measure*** is defined as

$$\Omega_{\boldsymbol{x}}^{\perp} = \int_D \underbrace{|\vec{\omega} \cdot \hat{n}(\boldsymbol{x})|}_{|\cos(\theta)|} \, \mathrm{d}\Omega(\vec{\omega}) \,, \tag{3.82}$$

where $D \subset \mathbb{S}^2(\boldsymbol{x})$, $\hat{n}(\boldsymbol{x})$ is the surface normal at $\boldsymbol{x}$, and θ is the angle between $\vec{\omega}$ and $\hat{n}(\boldsymbol{x})$. Geometrically, the projected solid angle can be understood as the area of the orthogonal projection of a solid angle onto the tangent plane at the surface (Figure 3.3).

Next, given a direction $\vec{\omega}$, we define the ***projected area measure*** as

$$A_{\vec{\omega}}^{\perp}(S) = \int_S \underbrace{|\vec{\omega} \cdot \hat{n}(\boldsymbol{x})|}_{|\cos(\theta)|} \, \mathrm{d}A(\boldsymbol{x}) \,, \tag{3.83}$$

where $S \subset \mathcal{M}$. The interpretation of this measure is to evaluate the projected area of a surface viewed from a particular direction (Figure 3.3). This measure corresponds to the projection of the surface onto a plane that is orthogonal to the direction $\vec{\omega}$.

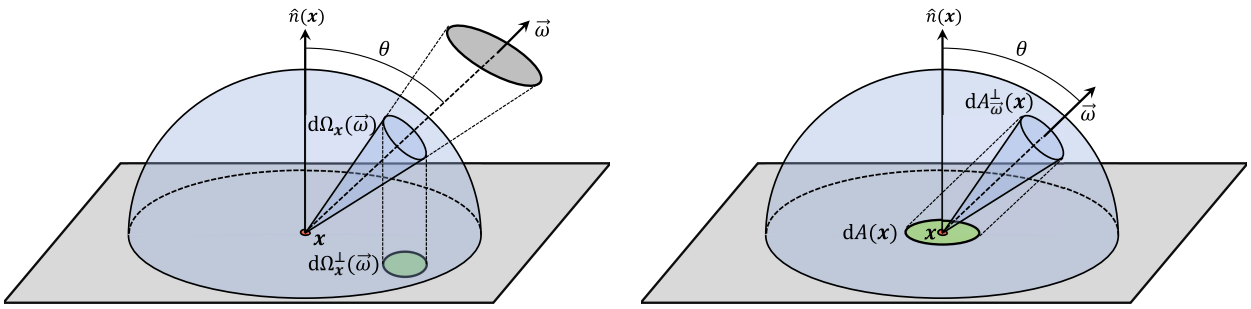


Figure 3.3: Common measures used in radiometry. (left) solid angle and projected solid angle measures, and (right) area and projected area measures.

With some abuse of notation, we can consider the differential version of these two measures to obtain the following relation

$$\mathrm{d}\Omega^{\perp}_{\boldsymbol{x}}(\vec{\omega})\,\mathrm{d}A(\boldsymbol{x}) = |\vec{\omega}\cdot\hat{n}(\boldsymbol{x})|\,\mathrm{d}\Omega_{\boldsymbol{x}}(\vec{\omega})\,\mathrm{d}A(\boldsymbol{x}) = \mathrm{d}\Omega_{\boldsymbol{x}}(\vec{\omega})\,\mathrm{d}A^{\perp}_{\vec{\omega}}(\boldsymbol{x})\,, \tag{3.84}$$

where we can absorb the cosine term into either of the original measures. This will be useful when defining radiance, as it allows us to integrate with respect to the simpler area and solid angle measures.

3.2.3 Radiometric Quantities

Radiometry is the branch that deals with the measurement of radiant energy transfer. To better understand the rendering equation and its meaning, it is useful to review some radiometric quantities and their relations. These will include the radiant energy, radiant power, irradiance and radiosity, and the radiance. Although all of these quantities can be defined in a measure-theoretic manner, we will instead drop a bit of rigor and use differential notations instead, as they are more intuitive on a physical level. The link between measures and differentials can be made slightly more rigorous through the Radon–Nikodym theorem [190, Chapter 3. Appendix B].

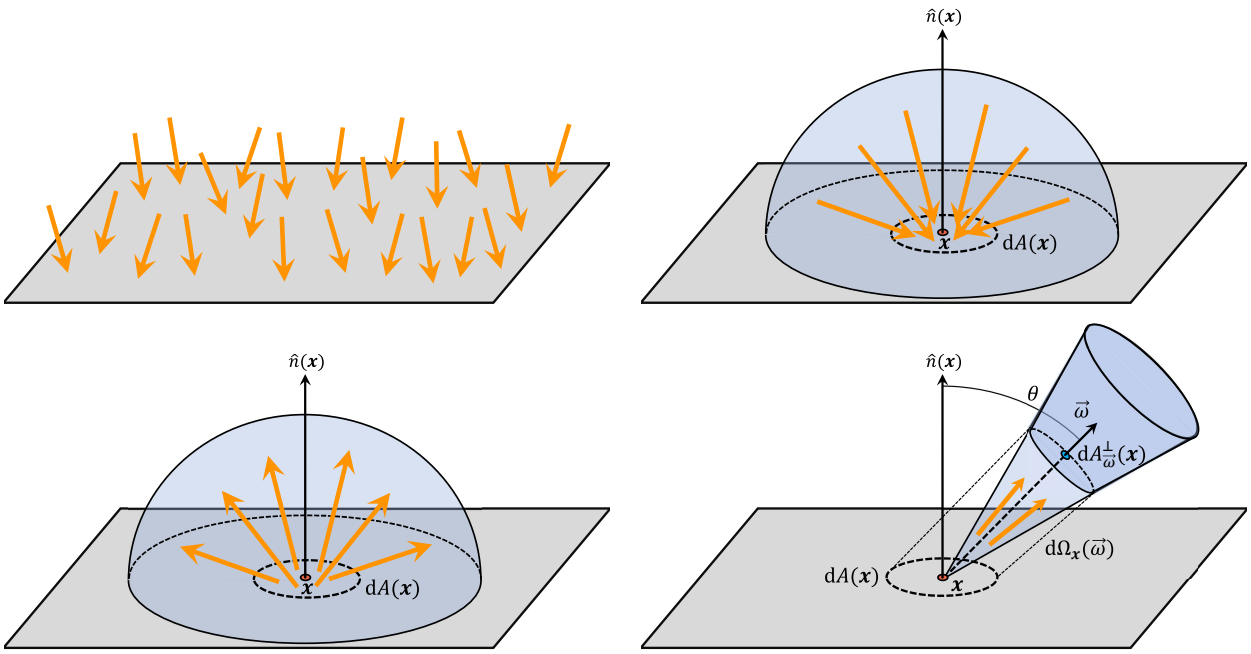


Figure 3.4: Common radiometric quantities. (Top-left) radiant power, (top-right) irradiance, (bottom-left) radiosity, and (bottom-right) radiance.

3.2.3.1 Radiant Energy

Denoted Q and measured in *joules* $[J]$, the ***(radiant) energy*** is the energy associated with electromagnetic radiation, such as photons. For a photon, it only depends on its specific wavelength λ through the relation

$$Q = \frac{hc}{\lambda}\,, \tag{3.85}$$

where $h = 6.626 \times 10^{-34}\ m^2\, kg\, s^{-1}$ is Planck's constant, and $c = 299,472,458\ m\, s^{-1}$ is the speed of light in a vacuum. It is the most basic radiometric quantity upon which all the other quantities presented here are built.

3.2.3.2 Radiant Power

In practice, we don't measure light one photon at a time and at a specific position in space. Physical measurements are typically done over a small time interval and surface. For this reason, it is more practical to work with quantities that represent such measurements.

This is where radiant power comes into play. Denoted by Φ and measured in *watts* [$W = J \cdot s^{-1}$], the ***(radiant) power***, also called ***(radiant) flux***, corresponds to the total amount of radiant energy Q_{tot} passing through a surface S over some time interval $[t_0, t_1]$ (Figure 3.4).

While photon streams do not consist of a continuum of particles, we can still approximate them as such, which enables us to write

$$Q_{tot} = \int_{t_0}^{t_1} \Phi(t) \, \mathrm{d}t \, . \tag{3.86}$$

While this makes the radiant power a function of time, we can rewrite it as

$$\Phi = \lim_{\Delta t \to 0} \frac{\Delta Q}{\Delta t} := \frac{\mathrm{d}Q}{\mathrm{d}t} \, , \tag{3.87}$$

since we are only interested in steady-state solutions. For this reason, we can consider Φ as a primitive quantity that is independent of time.

Note that we can restrict this quantity to measure only the amount of radiant energy that is *incident* on the surface, denoted as Φ_i, or the amount of energy *exiting* the surface, denoted as Φ_o. Similarly, we can consider the radiant power coming from a specific direction $\vec{\omega}$, which we denote as $\Phi(\vec{\omega})$.

3.2.3.3 Irradiance and Radiosity

Denoted as E, ***irradiance*** (also known as ***radiant power*** or ***flux density***) is the total amount of *incident* radiant power that *reaches* an infinitesimal patch around a surface point $\boldsymbol{x}$, with a surface normal $\hat{n}(\boldsymbol{x})$, per infinitesimal area $\mathrm{d}A(\boldsymbol{x})$ (Figure 3.4). It is defined by

$$E(\boldsymbol{x}) = \frac{\mathrm{d}\Phi_i(\boldsymbol{x})}{\mathrm{d}A(\boldsymbol{x})} \, . \tag{3.88}$$

Here, we have written $\Phi(\boldsymbol{x})$ to specify the radiant power over the infinitesimal area centered at $\boldsymbol{x}$ and to further emphasize its location dependence.

Similarly, ***radiosity***, also called ***radiant exitance*** and denoted by B, is the total amount of *exitant* radiant power that is *leaving* a surface per infinitesimal area. It is defined by

$$B(\boldsymbol{x}) = \frac{\mathrm{d}\Phi_o(\boldsymbol{x})}{\mathrm{d}A(\boldsymbol{x})}\,. \tag{3.89}$$

Like it is the case for radiant power, we can also consider the irradiance and radiance coming from a specific direction $\vec{\omega}$, which we denote as $E(\boldsymbol{x}, \vec{\omega})$ and $B(\boldsymbol{x}, \vec{\omega})$ respectively.

3.2.3.4 Radiance

Radiance, denoted as L, is undoubtedly the most important radiometric quantity in CG. It is, in fact, the most relevant quantity when it comes to the perceived brightness by the human eye. Radiance is defined by

$$L(\boldsymbol{x}, \vec{\omega}) = \frac{\mathrm{d}^2\Phi(\boldsymbol{x}, \vec{\omega})}{\mathrm{d}A^{\perp}_{\vec{\omega}}(\boldsymbol{x})\,\mathrm{d}\Omega_{\boldsymbol{x}}(\vec{\omega})}\,. \tag{3.90}$$

Intuitively, the radiance at $(\boldsymbol{x}, \vec{\omega})$ corresponds to the number of photons passing through an infinitesimal surface $\mathrm{d}A^{\perp}_{\vec{\omega}}(\boldsymbol{x})$ that is perpendicular to $\vec{\omega}$ and whose directions are contained in a small solid angle $\mathrm{d}\Omega_{\boldsymbol{x}}(\vec{\omega})$ around $\vec{\omega}$ during an infinitesimal amount of time (Figure 3.4). Using Equation (3.84), we can rewrite this definition as

$$L(\boldsymbol{x}, \vec{\omega}) = \frac{\mathrm{d}^2\Phi(\boldsymbol{x}, \vec{\omega})}{|\vec{\omega} \cdot \hat{n}(\boldsymbol{x})|\,\mathrm{d}\Omega_{\boldsymbol{x}}(\vec{\omega})\,\mathrm{d}A(\boldsymbol{x})} = \frac{\mathrm{d}^2\Phi(\boldsymbol{x}, \vec{\omega})}{\mathrm{d}\Omega^{\perp}_{\boldsymbol{x}}(\vec{\omega})\,\mathrm{d}A(\boldsymbol{x})}\,, \tag{3.91}$$

where $\hat{n}(\boldsymbol{x})$ is the surface normal at $\boldsymbol{x}$.

3.2.3.5 Incident and Exitant Radiance

Lastly, we need to define two other quantities that are derived from the radiance and depend on the direction $\vec{\omega}$. These are the ***incident radiance***, denoted as L_i, which measures light that is *arriving at* $\boldsymbol{x}$ ***from*** the direction $\vec{\omega}$, and the ***exitant radiance***,

denoted as L_o, which measures light that is *leaving from* $\boldsymbol{x}$ ***in*** direction $\vec{\omega}$. In free space, they are related by:

$$L_i(\boldsymbol{x}, \vec{\omega}) = L_o(\boldsymbol{x}, -\vec{\omega}) \,, \tag{3.92}$$

although they can, and will, differ on surfaces.

Note that we can also link the irradiance to the incident radiance by using Equation (3.84) and substituting Equation (3.88) in Equation (3.91):

$$\mathrm{d}E(\boldsymbol{x}, \vec{\omega}_i) = L_i(\boldsymbol{x}, \vec{\omega}_i)\, \mathrm{d}\Omega^{\perp}_{\boldsymbol{x}}(\vec{\omega}_i) \,. \tag{3.93}$$

3.2.4 Surface Light Transport

We are now prepared to explore the mechanics of light transport and the fundamental equations that govern the propagation of light through space and its interactions with objects. To do so, we will discuss the bidirectional scattering distribution function (BSDF) and the scattering equation, which define how radiance is affected by a surface material. We will also cover the measurement equation, which is basically what is computed by a camera sensor, and the many forms of the light transport equations up to the path integral equation.

3.2.4.1 Bidirectional Scattering Distribution Function

The ***BSDF***, denoted as f_s, is the mathematical model that describes how light scatters at a surface (Figure 3.5). It is defined as the observed radiance *leaving* in a direction $\vec{\omega}_o$, per unit of irradiance *arriving* from a direction $\vec{\omega}_i$ at a point $\boldsymbol{x} \in \mathcal{M}$:

$$f_s(\boldsymbol{x}, \vec{\omega}_i \to \vec{\omega}_o) = \frac{\mathrm{d}L_o(\boldsymbol{x}, \vec{\omega}_o)}{\mathrm{d}E(\boldsymbol{x}, \vec{\omega}_i)} = \frac{\mathrm{d}L_o(\boldsymbol{x}, \vec{\omega}_o)}{L_i(\boldsymbol{x}, \vec{\omega}_i)\, \mathrm{d}\Omega^{\perp}_{\boldsymbol{x}}(\vec{\omega}_i)} \,. \tag{3.94}$$

Here, we introduced the notation $\square \to \square$ to indicate the direction of light flow and used Equation (3.93). This relation is a consequence of the assumption that light behaves

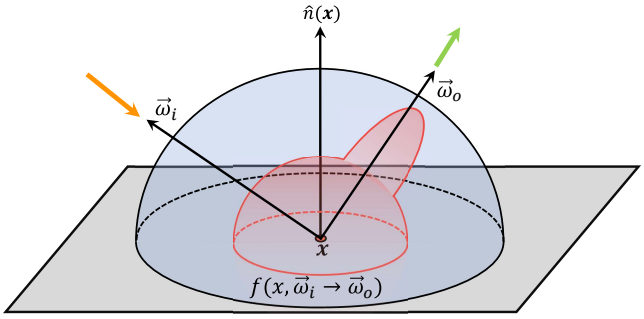


Figure 3.5: Bidirectional Scattering Distribution Function. The bidirectional scattering distribution function (BSDF) describes how radiance changes on a surface.

linearly and, therefore, a small increase in irradiance results in a linearly proportional increase in exiting radiance.

3.2.4.2 Surface Scattering Equation

We can integrate Equation (3.94) over all directions to obtain the ***surface scattering equation***:

$$L_o(\boldsymbol{x}, \vec{\omega}_o) = \int_{\mathbb{S}^2} L_i(\boldsymbol{x}, \vec{\omega}_i)\, f_s(\boldsymbol{x}, \vec{\omega}_i \rightarrow \vec{\omega}_o)\, \mathrm{d}\Omega^{\perp}_{\boldsymbol{x}}(\vec{\omega}_i)\,. \tag{3.95}$$

This equation is a fundamental ingredient for predicting the appearance of surfaces under illumination as it defines how the incoming radiance gets transformed by the surface material before and after a surface interaction.

3.2.4.3 Measurement Equation

In light transport, we are interested in creating an image from a scene description. To do so, we must evaluate the amount of incident radiance that is reaching each and every pixel from every possible direction. Additionally, we must account for the camera model, also called the sensor, while doing so. The camera model is usually given by a ***sensor responsivity*** function $W_e(\boldsymbol{x}, \vec{\omega})$ that scales the incident radiance $L_i(\boldsymbol{x}, \vec{\omega})$ according to the sensor properties. This is very similar to how a real-life camera sensor

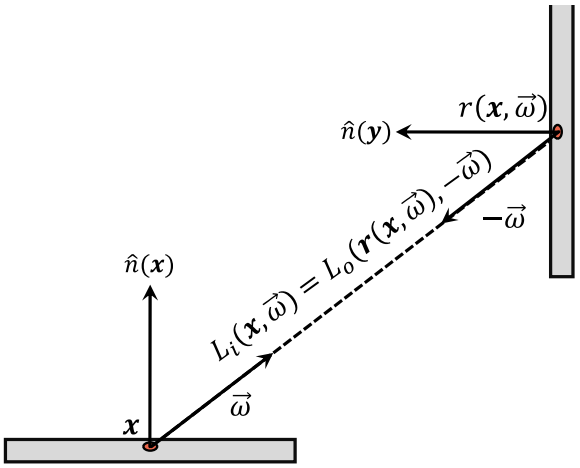


Figure 3.6: Ray casting. Illustration of ray casting and radiance conservation along that trajectory.

would work. This act of evaluating the effective pixel value modulated by the sensor responsitivity is called making a ***measurement***. Supposing that our image is composed of m pixels, this means that we want to perform m real-valued measurements, denoted $I_1, \ldots, I_m$. Mathematically, a measurement is done via the following integral

$$I_j = \int_{\mathcal{M} \times \mathbb{S}^2} W_e^j(\boldsymbol{x}, \vec{\omega})\, L_i(\boldsymbol{x}, \vec{\omega})\, \mathrm{d}\Omega_{\boldsymbol{x}}^{\perp}(\vec{\omega})\, \mathrm{d}A(\boldsymbol{x})\,. \tag{3.96}$$

This equation is called the ***measurement equation*** of the j-th pixel. Note that we integrated with respect to $\mathcal{M}$, but it would be more accurate to integrate with respect to the sensor surface. We assume that the sensor surface $\mathcal{S}$ is included in $\mathcal{M}$ and that W_e is non-zero only on the sensor. We make the same assumptions for the emitter surface $\mathcal{L}$ and their emitted radiance L_e. Down the line, this assumption will simplify the definition of the path integral formulation of the measurement equation.

3.2.4.4 Light Transport Equation

Since light is unaffected when traveling in free space and that we supposed that the radiance distribution has reached equilibrium, we know that the exiting radiance at

some point $\boldsymbol{x}$ leaving in one direction $\vec{\omega}$ will be conserved until it reaches the surface again. We define the function $r(\boldsymbol{x}, \vec{\omega})$, called the ***ray-casting function***, which returns the first point of $\mathcal{M}$ visible from $\boldsymbol{x}$ in the direction of $\vec{\omega}$ (Figure 3.6). Then, we know that the incident radiance at the new position $r(\boldsymbol{x}, \vec{\omega})$ will be the same as when it left $\boldsymbol{x}$, meaning

$$L_i(\boldsymbol{x}, \vec{\omega}) = L_o(r(\boldsymbol{x}, \vec{\omega}), -\vec{\omega})\,. \tag{3.97}$$

According to the law of conservation of energy and our initial assumptions, we know that the radiance leaving an object surface must come from somewhere — it is either emitted by the surface or coming from elsewhere and was scattered at this location. For this reason, we can then express the *total exitant radiance* L_o as the sum of an *emitted radiance* L_e and a *scattered radiance* $L_{o,s}$:

$$L_o = L_e + L_{o,s}\,. \tag{3.98}$$

In practice, the ***emitted radiance*** is given through the scene description. It represents all types of light sources in the scene, similar to how the sensor responsivity described the camera model. The ***scattered radiance*** $L_{o,s}$ corresponds to the total amount of in-scattered radiance that gets scattered and is given by Equation (3.95). Combining everything together results in the following equation

$$L_o(\boldsymbol{x}, \vec{\omega}_o) = L_e(\boldsymbol{x}, \vec{\omega}_o) + \underbrace{\int_{\mathbb{S}^2} L_i(\boldsymbol{x}, \vec{\omega}_i)\, f_s(\boldsymbol{x}, \vec{\omega}_i \to \vec{\omega}_o)\, \mathrm{d}\Omega^{\perp}_{\boldsymbol{x}}(\vec{\omega}_i)}_{L_{o,s}}\,. \tag{3.99}$$

Equivalently, we can rewrite this equation into a *recursive integral formula*:

$$L_o(\boldsymbol{x}, \vec{\omega}_o) = L_e(\boldsymbol{x}, \vec{\omega}_o) + \int_{\mathbb{S}^2} L_o(r(\boldsymbol{x}, -\vec{\omega}_i), -\vec{\omega}_i)\, f_s(\boldsymbol{x}, \vec{\omega}_i \to \vec{\omega}_o)\, \mathrm{d}\Omega^{\perp}_{\boldsymbol{x}}(\vec{\omega}_i)\,. \tag{3.100}$$

This form is what we call the ***light transport equation***. Since this equation does not depend on L_i anymore, it is common to drop the L_o notation and understand L to always correspond to the exitant radiance.

3.2.4.5 Importance Transport Equation

Recursively solving Equation (3.100) by starting from an emitter until a sensor is reached is called ***light tracing***. However, we can also derive an equation that does things the other way around using the adjoint formulation of light transport. Instead of following how radiance L propagates from emitter to sensor, we consider how a quantity called ***importance***, denoted as W, moves from a sensor to an emitter. The equation, called the ***importance transport equation***, describing this dynamic is given by

$$W(\boldsymbol{x}, \vec{\omega}_o) = W_e(\boldsymbol{x}, \vec{\omega}_o) + \int_{\mathbb{S}^2} W(r(\boldsymbol{x}, -\vec{\omega}_i), -\vec{\omega}_i)\, f_s(\boldsymbol{x}, \vec{\omega}_o \to \vec{\omega}_i)\, |\vec{\omega}_i \cdot \hat{n}(\boldsymbol{x})|\, \mathrm{d}\Omega_{\boldsymbol{x}}(\vec{\omega}_i)\,. \quad (3.101)$$

Here, W_e is now called the ***emitted importance function*** to better fit in this context. The only difference between Equation (3.100) and Equation (3.101) is the order of the incoming and exiting direction in the bidirectional scattering distribution function, which is not symmetric in the general case. Solving Equation (3.101) by starting from a sensor until an emitter is reached is called ***path tracing***. In general, light tracing methods are efficient when handling pointwise light sources with very glossy materials, and path tracing is efficient when handling pinhole cameras and non-glossy surfaces. More generally, they complement each other quite well, and modern methods, in fact, combine both approaches in what are called ***bidirectional path tracing (BDPT)*** methods [96, 191] (Figure 3.7).

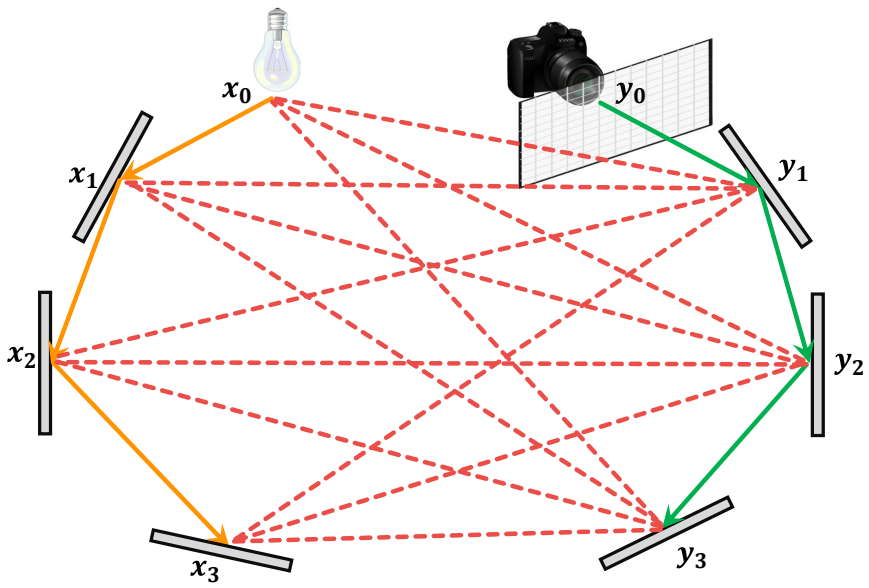


Figure 3.7: Types of path samplers. Path tracing (green) traces a path $\boldsymbol{y}_0\boldsymbol{y}_1\boldsymbol{y}_2\boldsymbol{y}_3\boldsymbol{x}_0$ from the camera toward the light source, light tracing (orange) traces a light path $\boldsymbol{x}_0\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3\boldsymbol{y}_0$ from a light source toward the camera, and bidirectional path tracing path tracing (red) does both, then connects everything together while averaging the contributions of all valid paths such as $\boldsymbol{x}_0\boldsymbol{x}_1\boldsymbol{y}_2\boldsymbol{y}_1\boldsymbol{y}_0$.

3.2.4.6 Three-Point Formulation

We can rewrite Equation (3.100) into a full area formulation that is more amenable to work with. To do so, we define the following notation

$$L(\boldsymbol{x} \to \boldsymbol{x}') = L(\boldsymbol{x}, \vec{\omega}) \,, \tag{3.102}$$

where $\vec{\omega} = \widehat{\boldsymbol{x}' - \boldsymbol{x}}$ is the unit vector pointing from $\boldsymbol{x}$ to $\boldsymbol{x}'$. Similarly, we write

$$W(\boldsymbol{x} \to \boldsymbol{x}') = W(\boldsymbol{x}', \vec{\omega}') \,, \tag{3.103}$$

where $\vec{\omega}' = \widehat{\boldsymbol{x} - \boldsymbol{x}'}$ is the unit vector pointing from $\boldsymbol{x}'$ to $\boldsymbol{x}$. Note that the direction of the arrow for the importance W is opposite to that of the radiance L since importance moves

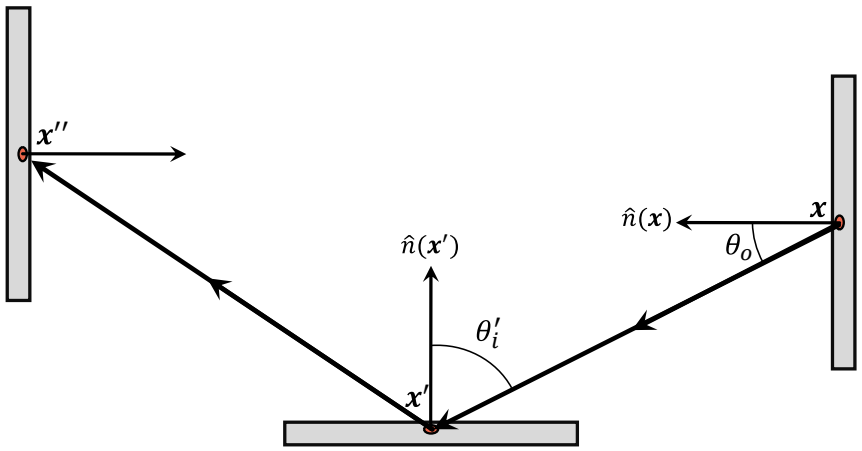


Figure 3.8: Geometry of the three-point formulation. This diagram shows the flow of light notation used in this section

in the opposite direction of light. Next, we write the BSDF as

$$f_s(\boldsymbol{x} \to \boldsymbol{x}' \to \boldsymbol{x}'') = f_s(\boldsymbol{x}', \vec{\omega}_i \to \vec{\omega}_o)\,, \tag{3.104}$$

where $\vec{\omega}_i = \widehat{\boldsymbol{x}' - \boldsymbol{x}}$ and $\vec{\omega}_o = \widehat{\boldsymbol{x}'' - \boldsymbol{x}'}$. To summarize, $\boldsymbol{x} \to \boldsymbol{x}'$ denotes the direction of light flow.

Then, we can also observe that we have the following geometric relation between the differential projected solid angle and differential area measure

$$\mathrm{d}\Omega^{\perp}_{\boldsymbol{x}'}(\vec{\omega}_i) = \mathrm{d}\Omega^{\perp}_{\boldsymbol{x}}(\widehat{\boldsymbol{x}' - \boldsymbol{x}}) = \frac{|\cos(\theta_o)\cos(\theta_i')|}{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}\,\mathrm{d}A(\boldsymbol{x})\,, \tag{3.105}$$

where $\theta_o = \angle\left(\widehat{\boldsymbol{x}' - \boldsymbol{x}}, \hat{n}(\boldsymbol{x})\right)$ and $\theta_i = \angle\left(\widehat{\boldsymbol{x} - \boldsymbol{x}'}, \hat{n}(\boldsymbol{x}')\right)$ are the polar angles between $\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$ and the surface normals at $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively, as shown in Figure 3.8. Combining everything together, we get the ***three-point form of the light transport equation***:

$$L(\boldsymbol{x}' \to \boldsymbol{x}'') = L_e(\boldsymbol{x}' \to \boldsymbol{x}'') + \int_{\mathcal{M}} L(\boldsymbol{x} \to \boldsymbol{x}')\, f_s(\boldsymbol{x} \to \boldsymbol{x}' \to \boldsymbol{x}'')\, G(\boldsymbol{x} \leftrightarrow \boldsymbol{x}')\,\mathrm{d}A\,, \tag{3.106}$$

where

$$G(\boldsymbol{x} \leftrightarrow \boldsymbol{x}') = V(\boldsymbol{x} \leftrightarrow \boldsymbol{x}') \frac{|\cos(\theta_o)\, \cos(\theta_i')|}{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}\,, \tag{3.107}$$

is called the ***geometry factor***, and

$$V(\boldsymbol{x} \leftrightarrow \boldsymbol{x}') = \begin{cases} 1 & r(\boldsymbol{x}, \widehat{\boldsymbol{x} - \boldsymbol{x}'}) = \boldsymbol{x}', \text{e.g., when } \boldsymbol{x}' \text{ and } \boldsymbol{x} \text{ are mutually visible}\,, \\ 0 & \text{otherwise}\,, \end{cases} \tag{3.108}$$

is the ***visibility function***, which is equal to 1 if there is no occlusion between $\boldsymbol{x}$ and $\boldsymbol{x}'$ and 0 otherwise. Finally, we can rewrite the measurement Equation (3.96) as

$$I_j = \int_{\mathcal{M}\times\mathcal{M}} W_e^j(\boldsymbol{x} \to \boldsymbol{x}')\, L(\boldsymbol{x} \to \boldsymbol{x}')\, G(\boldsymbol{x} \leftrightarrow \boldsymbol{x}')\, \mathrm{d}A(\boldsymbol{x})\, \mathrm{d}A(\boldsymbol{x}')\,. \tag{3.109}$$

To obtain the ***area form of the measurement equation***, note that we have switched the notation so that $\boldsymbol{x}'$ now represents the point on the sensor, while $\boldsymbol{x}$ represents the point defining the direction.

3.2.4.7 Path Integral Formulation

We are now ready to present the path integral formulation of light transport. Our goal is to write the measurement Equation (3.109) for the j-th pixel as

$$\int_{\mathcal{P}} f_j(\bar{\boldsymbol{x}})\, \mathrm{d}\mu_{\mathcal{P}}(\bar{\boldsymbol{x}})\,, \tag{3.110}$$

where we need to define some domain $\mathcal{P}$, an integrand f_j and a measure $\mu_{\mathcal{P}}$.

We define $\mathcal{P}^k$ as the ***set of all paths of length*** k, where $1 \leq k < \infty$. Elements of this space are all ***paths*** of the form $\bar{\boldsymbol{x}} = \boldsymbol{x}_0\boldsymbol{x}_1 \cdots \boldsymbol{x}_k$, where $\boldsymbol{x}_i \in \mathcal{M}$ (Figure 3.9). We can define a product measure $\mu_{\mathcal{P}^k}$ on this set of paths, called the ***area-product measure over paths of***

length k, as

$$\mu_{\mathcal{P}^k}(P_k) = \int_{P_k} \mathrm{d}A(\boldsymbol{x}_0) \cdots \mathrm{d}A(\boldsymbol{x}_k)\,, \tag{3.111}$$

where $P_k \in \mathcal{P}^k$ is a set of paths of length k. Building on this, we define the ***path space*** $\mathcal{P}$ to be the set of all paths of finite length:

$$\mathcal{P} = \bigcup_{k=1}^{\infty} \mathcal{P}^k\,. \tag{3.112}$$

We can define the ***path space area-product measure*** $\mu_{\mathcal{P}}$ as

$$\mu_{\mathcal{P}}(P) = \sum_{k=1}^{\infty} \mu_{\mathcal{P}^k}(P \cap \mathcal{P}^k)\,. \tag{3.113}$$

Here, $P \in \mathcal{P}$ is a set of paths of any finite length.

Returning to the last form of the measurement Equation (3.109), and recursively expanding the light transport Equation (3.100) inside it, we obtain

$$\begin{aligned} I_j &= \sum_{k=1}^{\infty} \int_{\mathcal{M}^{k+1}} L_e(\boldsymbol{x}_0 \to \boldsymbol{x}_1)\, G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1) \prod_{i=1}^{k-1} f_s(\boldsymbol{x}_{i-1} \to \boldsymbol{x}_i \to \boldsymbol{x}_{i+1})\, G(\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_{i+1}) \\ &\qquad \cdot W_e^j(\boldsymbol{x}_{k-1} \to \boldsymbol{x}_k)\, \mathrm{d}A(\boldsymbol{x}_0) \cdots \mathrm{d}A(\boldsymbol{x}_k) \\ &= \sum_{k=1}^{\infty} \int_{\mathcal{M}^{k+1}} f_j(\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)\, \mathrm{d}A(\boldsymbol{x}_0) \cdots \mathrm{d}A(\boldsymbol{x}_k) \\ &= \sum_{k=1}^{\infty} \int_{\mathcal{P}^k} f_j(\bar{\boldsymbol{x}}^k)\, \mathrm{d}\mu_{\mathcal{P}^k}(\bar{\boldsymbol{x}}^k)\,. \end{aligned} \tag{3.114}$$

This means that we can define the integrand separately for each path $\bar{\boldsymbol{x}}^k$ of length k, and it is given by (Figure 3.9)

$$\begin{aligned} f_j(\bar{\boldsymbol{x}}^k) = L_e(\boldsymbol{x}_0 \to \boldsymbol{x}_1)\, G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1) \prod_{i=1}^{k-1} f_s(\boldsymbol{x}_{i-1} \to \boldsymbol{x}_i \to \boldsymbol{x}_{i+1})\, G(\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_{i+1}) \\ \cdot W_e^j(\boldsymbol{x}_{k-1} \to \boldsymbol{x}_k)\,. \end{aligned} \tag{3.115}$$

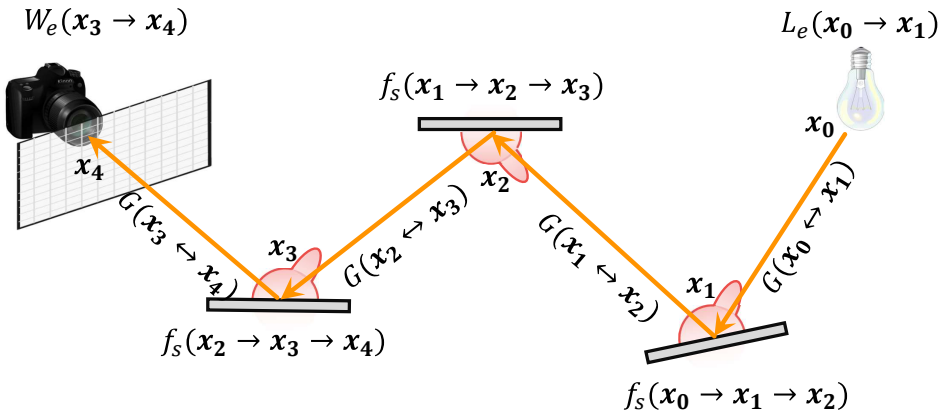


Figure 3.9: Path integral. Example of the path integral framework with a light path $\bar{\boldsymbol{x}}^4 = \boldsymbol{x}_0\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3\boldsymbol{x}_4$ of length 4.

Note that it is customary to factorize f_j, which is called the ***measurement contribution function***, into two separate functions. The first function h_j is called the ***reconstruction filter***, and the second one, f, is called the ***image contribution function*** or simply the path ***throughput***. These functions specify the parts that depend on the sensor and the parts that do not, respectively:

$$f_j(\bar{\boldsymbol{x}}^k) = \overbrace{L_e(\boldsymbol{x}_0 \to \boldsymbol{x}_1)\, G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1) \prod_{i=1}^{k-1} f_s(\boldsymbol{x}_{i-1} \to \boldsymbol{x}_i \to \boldsymbol{x}_{i+1})\, G(\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_{i+1})}^{f(\bar{\boldsymbol{x}}^k)} \cdot \underbrace{W_e^j(\boldsymbol{x}_{k-1} \to \boldsymbol{x}_k)}_{h_j(\bar{\boldsymbol{x}}^k)} . \tag{3.116}$$

Combining everything together, we obtain the final form of the ***path integral measurement equation***

$$I_j = \int_{\mathcal{P}} h_j(\bar{\boldsymbol{x}}) f(\bar{\boldsymbol{x}})\, \mathrm{d}\mu_{\mathcal{P}}(\bar{\boldsymbol{x}}) . \tag{3.117}$$

Now that we have this equation at hand, we can use MC methods to numerically solve it.

3.2.5 Monte Carlo Solutions to Light Transport

The objective of light transport algorithms is to calculate pixel values based on Equation (3.117). However, due to the complexity and high dimensionality of the integral, many rendering techniques rely on MCI for numerical approximation:

$$\langle I_j \rangle^N = \frac{1}{N} \sum_{i=1}^{N} \frac{h_j(\bar{\boldsymbol{x}}_i) f(\bar{\boldsymbol{x}}_i)}{p(\bar{\boldsymbol{x}}_i)} \,. \tag{3.118}$$

In order to approximate the integral using MCI, we need to sample independent paths $\bar{\boldsymbol{x}}_i$ according to some ***path probability density*** $p(\cdot)$, and evaluate their probability density. Several methods, such as path tracing, light tracing, and BDPT can be employed for this purpose. Without going into too much detail, all of these techniques rely on local path sampling strategies to sample new vertices. These strategies may involve directly sampling a direction within a pixel, sampling a new vertex on the emitter, sampling a new direction proportional to the bidirectional scattering distribution function, or reconnecting already existing vertices. Naturally, all local sampling strategies must be evaluated using the area measure. In the case of local directional sampling, adjustments must be made through a proper change of variables to return to the area measure, typically involving a Jacobian factor.

For example, in path tracing, we initially sample a direction in the pixel of interest and subsequently sample each following vertex according to the bidirectional scattering distribution function until the emitter is reached. MIS can also be utilized to consider the emitter at every stage, and next event estimation can be used to directly sample a point on the light source. The PDF is then the product of all local density functions used during sampling. With light tracing, the process is reversed. Both of these methods are instances of unidirectional path tracing. These methods can be computationally expensive, especially for scenes with complex lighting or small light sources, as it is challenging to find paths that connect the camera or these light sources by chance.

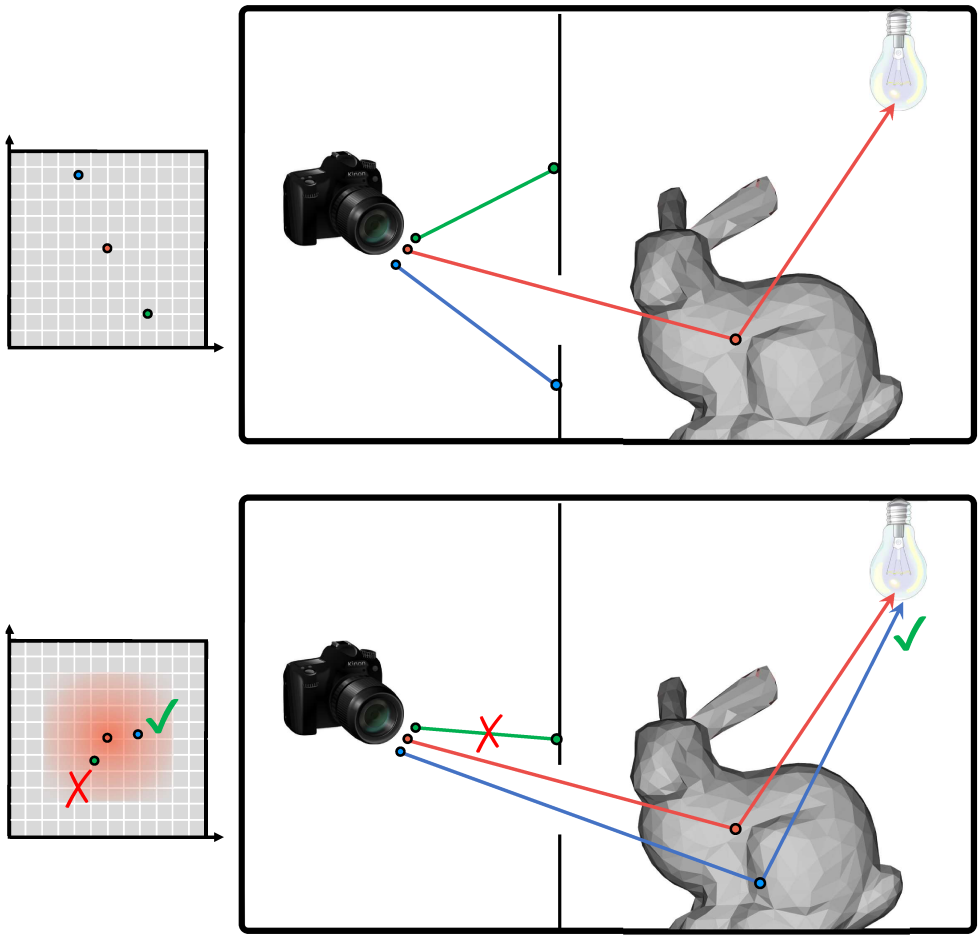


Figure 3.10: Monte Carlo ray tracing and Metropolis light transport. Monte Carlo ray tracing (top) samples a set of independent light paths, and Metropolis light transport (bottom) samples a sequence of correlated paths by mutating the current path and accepting the new path as the next path based on its contribution. We show the paths (right) in question and the primary samples (left) used to generate them to emphasize the differences.

BDPT combines both techniques by generating paths of length k by connecting s ***sensor-subpath*** vertices (also known as light-subpath) and t ***light-subpath*** vertices, where $k = s + t - 1$. These k (s, t) ***strategies*** are combined using MIS and appropriate reweighing of their distinct reconnection factors. By tracing paths from both the camera and the light sources, BDPT can more efficiently explore the space of light paths, especially in scenes with complex lighting or small light sources. This results in faster

convergence and reduced noise in the rendered image compared to unidirectional path tracing, even though the cost per sample is greater.

3.2.5.1 Metropolis Light Transport

Instead of generating paths independently, Veach and Guibas [193] proposed applying the MH algorithm to the path measurement Equation (3.117) in an algorithm called ***MLT*** (see Figure 3.10). In the MLT algorithm, the state space S is simply chosen as the path space $\mathcal{P}$, and the target distribution π is the scalar luminance of the path throughput $\tilde{f} = \ell \circ f$. This gives the following estimator

$$\langle I_j \rangle^N = \frac{Pb}{N} \sum_{i=1}^{N} \frac{h_j(\bar{\boldsymbol{x}}_i) f(\bar{\boldsymbol{x}}_i)}{\tilde{f}(\bar{\boldsymbol{x}})} \,. \tag{3.119}$$

Here, $b = \int_{\mathcal{P}} \tilde{f}(\bar{\boldsymbol{x}}) \, \mathrm{d}\mu_{\mathcal{P}}(\bar{\boldsymbol{x}})$ is the normalizing factor of the target $\tilde{f}$, P represents the number of pixels, and $\bar{\boldsymbol{x}}_i$ is a sequence of correlated samples generated through the MH acceptance probability using specially crafted transition kernels.

Of course, evaluating b is not easier than evaluating the full integral I_j for a specific pixel j. However, we only need to evaluate it once, and we can then reuse it for all other pixels. Usually, an unbiased estimation is obtained during a first pass using an MC estimator

$$\langle b \rangle^N = \frac{1}{N} \sum_{i=1}^{N} \frac{\tilde{f}(\bar{\boldsymbol{x}}_i)}{p(\bar{\boldsymbol{x}}_i)} \,. \tag{3.120}$$

At the same time, the initial state of the Markov chains is sampled from these precomputed paths $\bar{\boldsymbol{x}}_i$ by building a discrete distribution with the PMF set as the ***path contribution*** $C(\bar{\boldsymbol{x}}_i)$, which is defined as

$$C(\bar{\boldsymbol{x}}) = \frac{f(\bar{\boldsymbol{x}}_i)}{p(\bar{\boldsymbol{x}}_i)} \,. \tag{3.121}$$

This method avoids potential start-up bias that may arise from the initial states not being well distributed.

In the original method presented by Veach and Guibas [193], the transition kernel was a mixture of three types of ***local path mutations***, also called ***perturbations***, and ***global mutations***. The local mutations are the *lens mutations*, which are used to perturb paths containing specular surfaces visible from the camera, the *caustic mutations*, for paths exhibiting caustics visible from the camera, and the *multi-chain mutations* that perform both at the same time. The global mutation, which ensures ergodicity, is the *bidirectional mutation* and randomly replaces any subpath of the full path. Since then, many additional path space mutation strategies have been developed, such as *manifold exploration* [78], with Kaplanyan et al. [86] and Hanika et al. [70] later extending this approach to a more natural *half-vector reparameterization*, *geometry-aware* [145] and *specular manifold* sampling [204].

MLT is better than naive BDPT function in many aspects since it can reuse past information to explore interesting regions of the state space. For instance, it is better at handling scenes with difficult visibility and glossy caustics since it can start from a unique valid path and explore variations around it. However, while MLT typically results in estimators that exhibit less variance than BDPT, it is more prone to *fireflies* and *splotch artifacts* as chains can get stuck in narrow regions of high probability when the mutation strategy is not fitted to the target's local needs.

Primary Sample Space: MLT is a very powerful method to simulate light transport. However, it is quite difficult to implement. Many special cases must be handled while creating mutation strategies for different kinds of events. For this reason, it is quite challenging to craft general and efficient mutation strategies using this method. Primary sample space Metropolis light transport (PSSMLT) [89], on the other hand, relies on the fact that each path $\boldsymbol{x}$ is uniquely defined by a vector of random numbers $\bar{\boldsymbol{u}}$, called

primary samples, and a mapping function S that takes elements of this new space, called the primary sample space (PSS), and maps them to paths in the path space. We can construct this new space in a similar fashion than we did for the path space. First, we define the ***space of primary samples of length*** $O(k)$ $\mathcal{U}^k = [0,1]^{O(k)}$ as the hypercube of dimension $O(k)$, where $O(k)$ is the maximal number of random numbers used to generate a path of length k given any path sampler. This space possesses the natural measure $\mu_{\mathcal{U}^k}$, defined as the restriction of the Lebesgue measure of $\mathbb{R}^{O(k)}$ onto the hypercube. Then, we can define the ***PSS*** as

$$\mathcal{U} = \bigcup_{k=1}^{\infty} \mathcal{U}^k \,. \tag{3.122}$$

The ***PSS measure*** $\mu_{\mathcal{U}}$ is defined as

$$\mu_{\mathcal{U}}(U) = \sum_{k=1}^{\infty} \mu_{\mathcal{U}^k}(U \cap \mathcal{U}^k) \,, \tag{3.123}$$

where U is a set of primary samples. The mapping function $S : \mathcal{U} \to \mathcal{P}$ corresponds to the ***path sampling strategy***, also called the ***path sampler***, used to trace the path. By construction, it satisfies the following identity

$$p_S(S(\bar{\boldsymbol{u}})) = \left| \frac{\mathrm{d}\mu_{\mathcal{P}}(\bar{\boldsymbol{x}})}{\mathrm{d}\mu_{\mathcal{U}}(\bar{\boldsymbol{u}})} \right|^{-1} \,, \tag{3.124}$$

where p_S is the probability density of the sampling technique S, and the right term is the Jacobian determinant of the change of variable. This change of variable results in the

primary sample measurement equation:

$$\begin{aligned} I_j &= \int_{\mathcal{P}} h_j(\bar{\boldsymbol{x}}) f(\bar{\boldsymbol{x}}) \, \mathrm{d}\mu_{\mathcal{P}}(\bar{\boldsymbol{x}}) \\ &= \int_{\mathcal{U}} h_j(S(\bar{\boldsymbol{u}})) f(S(\bar{\boldsymbol{u}})) \left| \frac{\mathrm{d}\mu_{\mathcal{P}}(\bar{\boldsymbol{x}})}{\mathrm{d}\mu_{\mathcal{U}}(\bar{\boldsymbol{u}})} \right| \mathrm{d}\mu_{\mathcal{U}}(\bar{\boldsymbol{u}}) \\ &= \int_{\mathcal{U}} h_j(S(\bar{\boldsymbol{u}})) \frac{f(S(\bar{\boldsymbol{u}}))}{p_S(S(\bar{\boldsymbol{u}}))} \, \mathrm{d}\mu_{\mathcal{U}}(\bar{\boldsymbol{u}}) \\ &= \int_{\mathcal{U}} h_j(S(\bar{\boldsymbol{u}})) C_S(\bar{\boldsymbol{u}}) \, \mathrm{d}\mu_{\mathcal{U}}(\bar{\boldsymbol{u}}) \,, \end{aligned} \tag{3.125}$$

where we used the path contribution notation defined earlier:

$$C_S(\bar{\boldsymbol{u}}) = \frac{f(S(\bar{\boldsymbol{u}}))}{p_S(S(\bar{\boldsymbol{u}}))} \,. \tag{3.126}$$

This means that in this new space, the natural throughput is the path contribution. In general, PSSMLT relies on BDPT and uses MIS to incorporate all the strategy contributions:

$$C(\bar{\boldsymbol{u}}) = \sum_{m=1}^{M} w_m(S_m(\bar{\boldsymbol{u}})) \frac{C_m(S_m(\bar{\boldsymbol{u}}))}{p_m(S_m(\bar{\boldsymbol{u}}))} \,, \tag{3.127}$$

where M is the total number of sampling strategies available, m refers to the m-th path sampling strategy S_m, and the w_m are the MIS weights.

In PSSMLT, the state space S is chosen as the PSS $\mathcal{U}$, and the target distribution π is the scalar luminance of the contribution functions $\tilde{C} = \ell \circ C$. This gives the following estimator

$$\langle I_j \rangle^N = \frac{P b^*}{N} \sum_{i=1}^{N} \frac{h_j(m(\bar{\boldsymbol{u}}_i)) C(\bar{\boldsymbol{u}}_i)}{\tilde{C}(\bar{\boldsymbol{u}}_i)} \,, \tag{3.128}$$

where $b^* = \int_{\mathcal{U}} \tilde{C}(\bar{\boldsymbol{u}}) \, \mathrm{d}\mu_{\mathcal{U}}(\bar{\boldsymbol{u}})$ is the normalizing factor of the target $\tilde{C}$, and P is the number of pixels.

Working directly in the PSS has several advantages. Firstly, the target distribution is smoother than that of MLT because it also considers the path probability which is mostly proportional to the path throughput except for visibility, making it easier to explore.

Additionally, creating general mutation strategies on the hypercube is easier. In their original paper, Kelemen et al. [89] provided an isotropic *small step mutation* strategy that mutates all components individually. This perturbation relies on a distribution with an exponential falloff centered at the current state with a small hole at its center to encourage exploration. In addition, they provide a *large step mutation* that ensures ergodicity by sampling new points uniformly over the hypercube. However, it is difficult to predict the effect that a small mutation in the PSS will have in the path space due to lack of geometric information. Many authors have proposed alternative solutions to this problem. For instance, Otsu et al. [144] and Pantaleoni [147] convert path space mutation strategies to their PSS representations, enabling path space-like mutations directly in PSS. Similarly, Bitterli et al. [14] employ *reversible jump* to map random numbers to different strategies without altering the path geometry.

Alternatively, anisotropic Gaussian mutations have been proposed by Li et al. [101]. This method leverages automatic differentiation and Hamiltonian Monte Carlo (HMC) to better fit the target density by computing the gradient and Hessian of the log-target. More recently, Luan et al. [109] developed a mutation strategy combining the Metropolis adjusted Langevin algorithm (MALA) with momentum schemes from stochastic gradient descent procedures, providing another anisotropic Gaussian mutation with drift.

Multiplexed: When used in combination with BDPT, both MLT and PSSMLT generate a large number of paths, many of which may contribute little to no energy to the final image. To address this problem, multiplexed Metropolis light transport (MMLT) [67] introduces an extra dimension $\mathcal{U}_m = [0, 1] \times \mathcal{U}$ to the PSS, allowing the chains to explore regions where certain path sampling methods outperform others. This approach eliminates the need to compute all visibility connection factors of other strategies, effectively exploring only with the optimal strategy.

Additionally, in MMLT, the target distribution considers only one sampling strategy S_m at a time and includes the MIS weight w_m within the throughput:

$$C_m(\bar{\boldsymbol{u}}) = w_m(S_m(\bar{\boldsymbol{u}}))\frac{f(S_m(\bar{\boldsymbol{u}}))}{p_m(S_m(\bar{\boldsymbol{u}}))}\,. \tag{3.129}$$

This gives us the following estimator

$$\langle I_j\rangle^N = \frac{MPb^*}{N}\sum_{i=1}^{N}\frac{h_j(m(\bar{\boldsymbol{u}}_i))C_m(\bar{\boldsymbol{u}}_i)}{\tilde{C}_m(\bar{\boldsymbol{u}}_i)}\,, \tag{3.130}$$

where $\tilde{C}_m = \ell \circ C_m$ is the target function, and $b^* = \int_{\mathcal{U}} \tilde{C}(\bar{\boldsymbol{u}})\,\mathrm{d}\mu_{\mathcal{U}}(\bar{\boldsymbol{u}})$ is the same normalizing factor as PSSMLT. Note that we need to scale by the number of techniques M in addition to the number of pixels P to account for the fact that we explore one strategy at the time.

Note that all the mutation strategies described in Section 3.2.5.1 can also be used in MMLT, in addition to a new ***multiplexed mutation*** strategy that perturbs the multiplexed component to change the sampling technique.

3.3 Fundamentals of Fluid Dynamics

In this section, we will review key concepts in fluid dynamics, such as the incompressible NS equations, the vorticity transport equation, and discretization methods. For further information on this subject, we encourage readers to consult the book from Bridson [18].

3.3.1 Incompressible Navier–Stokes Equations

In CG, most fluid simulations are based on the ***incompressible NS equations***, which are a set of PDEs that describe the motion of incompressible fluids. These equations can be

expressed as follows:

$$\overbrace{\frac{D\vec{v}}{Dt}}^{\text{Advection}} = \overbrace{-\frac{1}{\rho}\nabla p}^{\text{Internal pressure}} + \overbrace{\nu\nabla^2\vec{v}}^{\text{Diffusion}} + \overbrace{\vec{f}}^{\text{External forces}} , \tag{3.131}$$

$$\underbrace{\nabla\cdot\vec{v}=0}_{\text{Incompressibility}} . \tag{3.132}$$

Here, $\vec{v}$ represents the ***velocity*** field, which describes the flow of the fluid at every point in space. p denotes the internal ***pressure***, which is the pressure exerted by the fluid on its surroundings. ρ is the fluid ***density***, which measures the amount of mass per unit volume of the fluid. ν corresponds to the ***kinematic viscosity***, which describes the resistance of the fluid to deformation. Finally, $\vec{f}$ contains all ***external forces*** acting on the fluid, such as gravity or fluid-solid interactions. Furthermore, $D/Dt = \partial/\partial t + (\vec{v}\cdot\nabla)$ is called the ***material derivative***, which describes the rate of change of a physical quantity along the fluid's motion. The first equation is also called the ***momentum equation*** and the second one the ***continuity equation***. The continuity equation enforces mass conservation, while the momentum equation describes how the fluid's momentum changes over time due to pressure gradients, viscous forces, and external forces.

3.3.1.1 Initial and Boundary Conditions

Since the incompressible NS equations are a set of spatiotemporal PDEs, additional conditions are necessary to uniquely define a solution. Firstly, an ***initial condition*** is required to resolve the temporal ambiguity. This is typically achieved by prescribing the velocity field at a specific time of interest, such as $t = 0$. Next, the spatial extra degree of freedom must be addressed. The incompressible NS equations describe the fluid behavior within a domain Ω. However, when this domain is bounded, the fluid behavior at the boundary $\partial\Omega$ of the domain must be specified. The most common ***boundary conditions*** encountered in fluid simulations include:

- **Free-slip:** The free-slip boundary condition is applied by setting the normal component of the velocity field to zero at the boundary, while the tangential component remains unconstrained. This results in a flow that is parallel to the boundary surface, with no fluid passing through it. The free-slip condition is often used in cases where the fluid interaction with the boundary is negligible or when the computational cost of more complex boundary conditions is prohibitive.
- **No-slip:** The no-slip boundary condition enforces that the velocity of the fluid at the boundary is equal to the velocity of the boundary itself. In most cases, this means that the fluid velocity is set to zero at the boundary, simulating the effect of viscous friction between the fluid and the surface. The no-slip condition is widely observed in real-world fluid flows, particularly in flows with high viscosity or when the fluid comes into contact with rough or textured surfaces. The energy dissipation at the boundary in no-slip conditions often leads to the formation of a boundary layer, where the fluid velocity changes rapidly with distance from the boundary.
- **Periodic:** Periodic boundary conditions are useful when simulating flows in periodic structures or when a small section of a larger flow field is of interest. By assuming that the domain repeats itself in space, the computational cost can be reduced while still capturing the essential features of the flow. To implement periodic boundary conditions, the fluid properties, such as velocity and pressure, are set to be equal at the corresponding points on opposite sides of the domain. This effectively creates a seamless transition between the opposite boundaries, simulating an infinite, repeating flow.

3.3.2 Vorticity Transport Equation

The vorticity form of the incompressible NS equations is an alternative representation that emphasizes the role of vorticity in the fluid flow (Figure 2.7). ***Vorticity***, denoted as

$\vec{\omega}$, is a vector field defined as the curl of the velocity field, $\vec{v}$, which can be written as

$$\vec{\omega} = \nabla \times \vec{v}\,. \tag{3.133}$$

By taking the curl of the incompressible NS equations given in Equation (3.131), we obtain the ***vorticity transport equation***:

$$\frac{D\vec{\omega}}{Dt} = \mathcal{D}_{\vec{v}}\vec{\omega} + \nu\nabla^2\vec{\omega} + \nabla \times \vec{f}, \tag{3.134}$$

$$\vec{\omega} = \nabla \times \vec{v}, \tag{3.135}$$

where $\vec{\omega}$ is the vorticity, $\nabla\times$ is the curl operator and $\mathcal{D}_{\vec{v}} = (\nabla\vec{v} + \nabla\vec{v}^{\intercal})/2$ is called the ***strain rate tensor***. Here,$\mathcal{D}_{\vec{v}}$ is an alternative, but equivalent form, to the ***stretching term*** $(\vec{\omega} \cdot \nabla)\vec{v}$ that is easier to work with. Both of these forms have advantages and disadvantages depending on the use case. Here are a few:

+ **Vorticity preservation:** The vorticity transport equation focuses on the vorticity field, making it particularly suitable for problems where the preservation of vorticity structures is crucial. This can be important in capturing visually interesting and physically accurate fluid motions in CG and other applications.
+ **Decoupling of pressure and velocity:** In the vorticity transport equation, the pressure and velocity fields are decoupled, which can simplify the numerical solution process. This decoupling can also help avoid the need for solving a pressure Poisson equation, which is often required in NS-based methods.
− **Reconstruction of the velocity field:** In vorticity-based methods, the velocity field must often be reconstructed from the vorticity field, which can introduce additional computational complexity and may be less accurate than directly solving for the velocity field using the NS equations.
− **Boundary conditions:** Incorporating boundary conditions in vorticity-based simulations can be more challenging than in NS-based simulations. This can lead to

difficulties when dealing with complex geometries or various types of boundary conditions.

3.3.2.1 Biot–Savart Law

It is possible to compute the velocity from the vorticity using the ***BS law*** [34]:

$$\vec{v}(\boldsymbol{x}) = \int_{\Omega} \vec{\omega}(\boldsymbol{y}) \times G(\boldsymbol{x} - \boldsymbol{y}) \, \mathrm{d}\mu(\boldsymbol{y}) \, . \tag{3.136}$$

Here, μ is the appropriate Lebesgue measure of the integration domain Ω, and G is called the ***BS kernel***, which is related to Green's function, and is given by

$$G(\boldsymbol{x} - \boldsymbol{y}) = \begin{cases} \dfrac{1}{2\pi} \dfrac{\boldsymbol{x} - \boldsymbol{y}}{\|\boldsymbol{x} - \boldsymbol{y}\|^2} & \text{in 2D} \, , \\ \dfrac{1}{4\pi} \dfrac{\boldsymbol{x} - \boldsymbol{y}}{\|\boldsymbol{x} - \boldsymbol{y}\|^3} & \text{in 3D} \, . \end{cases} \tag{3.137}$$

Note that these kernels diverge when $\boldsymbol{x} = \boldsymbol{y}$, which can lead to numerical instabilities when evaluating the BS integral (3.136). To tackle this problem, we can use the ***Rosenhead-Moore kernel approximation*** instead [117] to remove the singularity:

$$G_{\delta}(\boldsymbol{x} - \boldsymbol{y}) = \begin{cases} \dfrac{1}{2\pi} \dfrac{\boldsymbol{x} - \boldsymbol{y}}{\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \delta^2} & \text{in 2D} \, , \\ \dfrac{1}{4\pi} \dfrac{\boldsymbol{x} - \boldsymbol{y}}{(\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \delta^2)^{\frac{3}{2}}} & \text{in 3D} \, , \end{cases} \tag{3.138}$$

where δ is a user-set smoothing parameter.

3.3.3 Discretization

There are two main approaches to discretize the incompressible NS and vorticity transport equations: Eulerian and Lagrangian (Figure 2.6). ***Eulerian*** methods are a class of numerical techniques used to simulate fluid flow and other dynamic systems by focusing on *fixed points in space*. They describe the evolution of fluid properties, such as

velocity, pressure, vorticity, and density at these fixed spatial locations as the fluid flows through them. ***Lagrangian*** methods are a class of numerical techniques used to simulate fluid flow by focusing on *individual fluid particles* as they move through space. They describe the evolution of fluid properties, such as velocity, pressure, vorticity, and density by tracking the motion of these particles over time. Typically, Eulerian methods are associated with *grid and mesh-based methods*, while Lagrangian methods are related to *particle and segment-based methods*. However, it is more appropriate to understand them based on *how the quantities are stored and advected in time*. For instance, an Eulerian approach could use a fixed grid representation at every time step, updating the fluid properties at these fixed locations. Both Eulerian and Lagrangian methods have their advantages and limitations. Eulerian methods can handle complex geometries and boundary conditions more easily, but they can suffer from numerical diffusion and may struggle to capture fine details in the flow. Lagrangian methods can provide more accurate results by following the fluid motion, but they can be computationally expensive and may struggle with complex geometries or boundary conditions. It is worth noting that ***hybrid methods*** have been developed to combine the strengths of both Eulerian and Lagrangian approaches. These methods aim to leverage the advantages of each approach while mitigating their respective limitations, providing more accurate and efficient solutions for simulating fluid flows.

3.3.3.1 Semi-Lagrangian Method

In our work, we will rely on an advection scheme called ***semi-Lagrangian advection*** (Figure 3.11). This advection method was first proposed in fluid animation by Stam [178]. The key idea of the semi-Lagrangian method is to track the quantity of interest along the fluid's trajectories, which are dictated by the velocity field $\vec{v}$. This approach is called semi-Lagrangian because *it combines the Eulerian description of the fixed field variables*

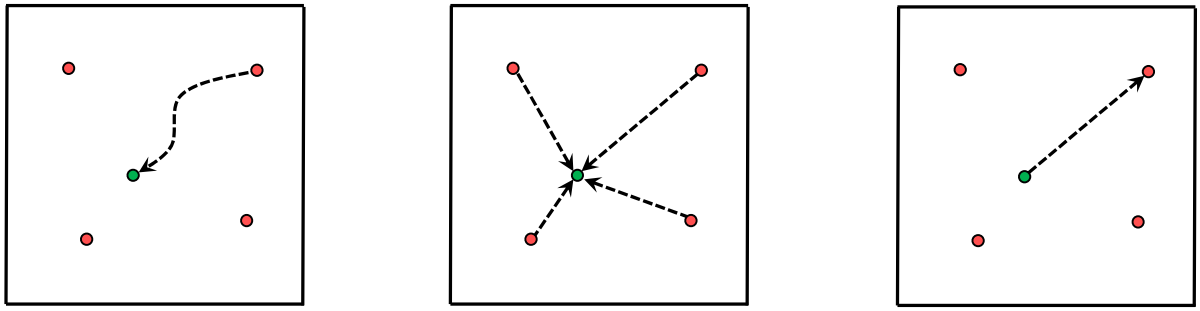


Figure 3.11: Semi-Lagrangian Method. Steps of the semi-Lagrangian Method: (left) we trace the position of interest trajectory backward in time to get its previous location, (middle) we interpolate from the previous values known at fixed locations, and (right) we update the quantity of interest based on the interpolated value. Here, the fixed known locations are not aligned on a grid, deliberately demonstrating that this alignment is not a necessity.

with the Lagrangian tracking of individual particles. The semi-Lagrangian advection method involves the following steps:

- **Trace:** Trace the position of interest (particle) trajectory backward in time to find its previous location, also called departure point, using the velocity field. This can be done using various numerical integration methods such as Euler or Runge-Kutta.
- **Interpolate:** Use interpolation to estimate the value of the quantity of interest at the departure point. Interpolation methods, such as linear, cubic, or higher-order schemes, can be used to estimate the field values at unknown spatial locations. Note that this can be done with values stored on a regular grid or, more generally, at fixed unstructured positions in space.
- **Update:** Assign the interpolated value of the quantity of interest at the departure point to the current position, effectively advecting the scalar quantity forward in time.

Of course this method also comes with advantages and disadvantages depending on the use case. Here are a few:

+ **Stability:** It is unconditionally stable, meaning it does not suffer from the restrictive time step limitations imposed by the Courant-Friedrichs-Lewy condition, which is common in other advection schemes.
+ **Conservation:** By design, semi-Lagrangian advection conserves the transported scalar quantity along the flow.
− **Diffusion:** Due to the interpolation step, the method can introduce numerical diffusion, leading to a loss of small-scale details in the solution.
− **Cost:** The need to trace particle trajectories and interpolate values can make the method more computationally expensive compared to simpler explicit Eulerian advection schemes due to the extra computations.

As mentioned, semi-Lagrangian advection can be combined with various numerical integration methods to trace particle trajectories more accurately. Two such methods are the Euler method and the Runge–Kutta 4th order (RK4) method. Here, we will discuss how these two methods can be used within the semi-Lagrangian framework.

Euler: The simplest numerical integration method is the ***Euler method***. When used within the semi-Lagrangian advection framework, it computes the departure point for each grid point by tracing the particle trajectory backward in time using a single step:

$$\boldsymbol{x}' = \boldsymbol{x} - \Delta t\, \vec{v}(\boldsymbol{x}, t)\,, \tag{3.139}$$

where $\boldsymbol{x}'$ is the departure position, $\boldsymbol{x}$ is the current position, Δt is the time step, and $\vec{v}$ is the velocity field. The Euler method is computationally efficient but can introduce inaccuracies when dealing with rapidly changing or highly nonlinear velocity fields.

Runge–Kutta 4: The ***RK4 method*** is a higher-order numerical integration method that provides improved accuracy compared to the Euler method. In the context of semi-Lagrangian advection, the RK4 method computes the departure point by taking

multiple intermediate steps:

$$\vec{k}_1 = \vec{v}\left(\boldsymbol{x}, t\right) \tag{3.140}$$

$$\vec{k}_2 = \vec{v}\left(\boldsymbol{x} - \frac{\Delta t}{2}\,\vec{k}_1, t - \frac{\Delta t}{2}\right) \tag{3.141}$$

$$\vec{k}_3 = \vec{v}\left(\boldsymbol{x} - \frac{\Delta t}{2}\,\vec{k}_2, t - \frac{\Delta t}{2}\right) \tag{3.142}$$

$$\vec{k}_4 = \vec{v}\left(\boldsymbol{x} - \Delta t, \vec{k}_3, t - \Delta t\right) \text{ and} \tag{3.143}$$

$$\boldsymbol{x}' = \boldsymbol{x} - \frac{\Delta t}{6}\,\left(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4\right)\,. \tag{3.144}$$

While the RK4 method is more computationally expensive than the Euler method, it provides significantly improved accuracy, making it a popular choice for semi-Lagrangian advection in situations where the velocity field exhibits complex behavior or high nonlinearity.

Chapter 4

Delayed Rejection Metropolis Light Transport (DRMLT)

We tackle the challenge of creating flexible mutation strategies within Metropolis light transport (MLT). The method presented here is based on our publication [157]. This paper was published in Association for Computing Machinery (ACM) Transactions on Graphics (TOG) and presented at the ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH) North America 2020 conference. For more information on my contribution to this work, please refer to Section 1.2.1.

Damien Rioux-Lavoie, Joey Litalien, Adrien Gruson, Toshiya Hachisuka, and Derek Nowrouzezahrai. Delayed rejection Metropolis light transport. *ACM Transactions on Graphics (TOG)*, 39(3):1–14, 2020

4.1 Introduction

Photorealistic rendering simulates the physics of light propagation according to scattering interactions between emitters and surfaces. The radiometric dynamics of light

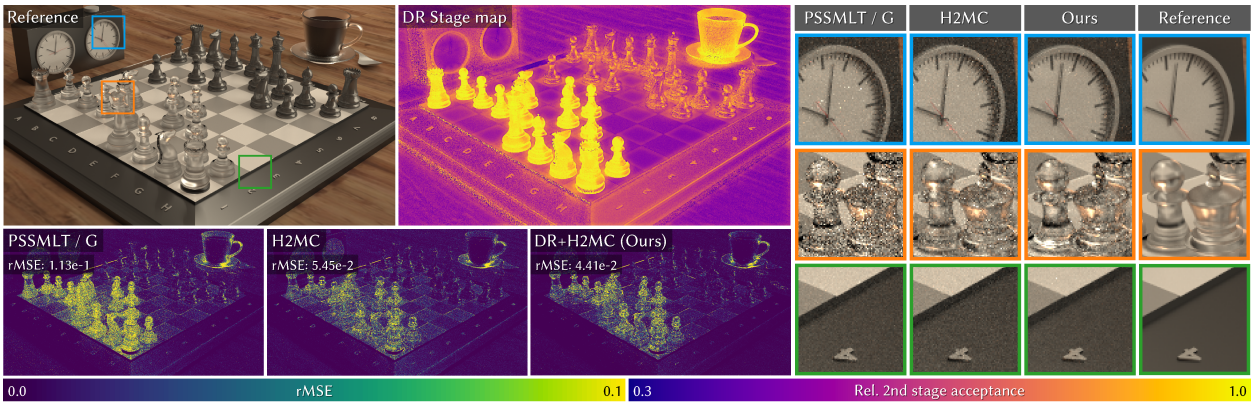


Figure 4.1: DRMLT. We generalize the Metropolis–Hastings algorithm with DR [188, 58]: our *delayed rejection Metropolis light transport (DRMLT)* method selectively applies different mutation strategies, improving upon one-stage primary sample space algorithms, i.e., PSSMLT [89] with Gaussian proposals (PSSMLT / G) and H2MC [101]. One variant of our method first attempts an isotropic Gaussian proposal, resorting to more intricate kernels (that improve local exploration with differential information) only when the first attempt failed, e.g., on rough dielectrics. DRMLT focuses computations in hard-to-explore regions without compromising quality in comparatively simpler regions (e.g., on the board). We visualize a per-pixel relative second-stage acceptance, where violet and yellow extremes respectively indicate the efficiency of the first and second stages.

transport can become arbitrarily complex, especially in scenes with physically-based materials and detailed geometry, leading to many numerical challenges. Here, even state-of-the-art Monte Carlo (MC) methods can fall short of accurately and efficiently integrating complicated transport.

Markov chain Monte Carlo (MCMC) methods (Section 3.1.4), introduced to rendering by Veach and Guibas [193] with their seminal MLT algorithm (Section 3.2.5.1), evolves a sequence of correlated light paths in a manner that guarantees convergence to the target radiometric density. MLT applies the Metropolis–Hastings (MH) algorithm (Section 3.1.4.1) [122, 72] to the path integral [190] (Section 3.2.4.7)—an integral over *path space*, i.e., the infinite-dimensional space of light transport paths. The statistical dependence intrinsic to Markov chains allows MLT-based techniques to locally explore

high-contribution regions in path space, as they are encountered, by sequentially perturbing the path structure according to *mutation strategies*.

Despite advances in MCMC light transport [175, 12], designing efficient, robust and general-purpose mutation strategies remains an open problem. Smaller perturbations can improve acceptance rates, but not without often slowing the exploration of the salient regions in state space. Alternatively, larger perturbations may reduce correlation between samples, but these proposals are often rejected as they tend to escape high density regions, leading to chains getting "trapped" and variance increases in the resulting estimator [5]. Another form of this trade-off appears when considering the run-time cost of transitions: intricate mutation strategies can be more effective at equal sample count while typically requiring more computations; applying such costly transitions everywhere is wasteful in simple scenarios. However, completely abandoning them could lead to under-exploration of important regions. Our work focuses on designing efficient, flexible proposal mechanisms in such cases.

We present a two-stage proposal mechanism that automatically balances local exploration and computational efficiency. Our approach extends *delayed rejection (DR)* [188, 58], a method for sequentially combining many transition kernels in order to propose more robust perturbations. DR exploits prioritization by proposing bolder, i.e. a large and more aggressive, or less costly transitions at a first stage before falling-back to more timid or expensive kernels upon failure (Figure 4.2). By augmenting proposal mechanisms with such a *safety net*, our method can increase acceptance rates without compromising exploration of the state space, especially in complex path sampling scenarios. A naïve implementation of DR can lead to zero-acceptance in the second stage with a Markov kernel lacking global support (e.g., *Kelemen-style* mutations [89]). We thus introduce a novel combination of proposals to address this problem: our *pairwise orbital* mutation conditions on both the current and first rejected states ensure that all states remain close to each other, thus eliminating degenerate cases that can drag acceptance

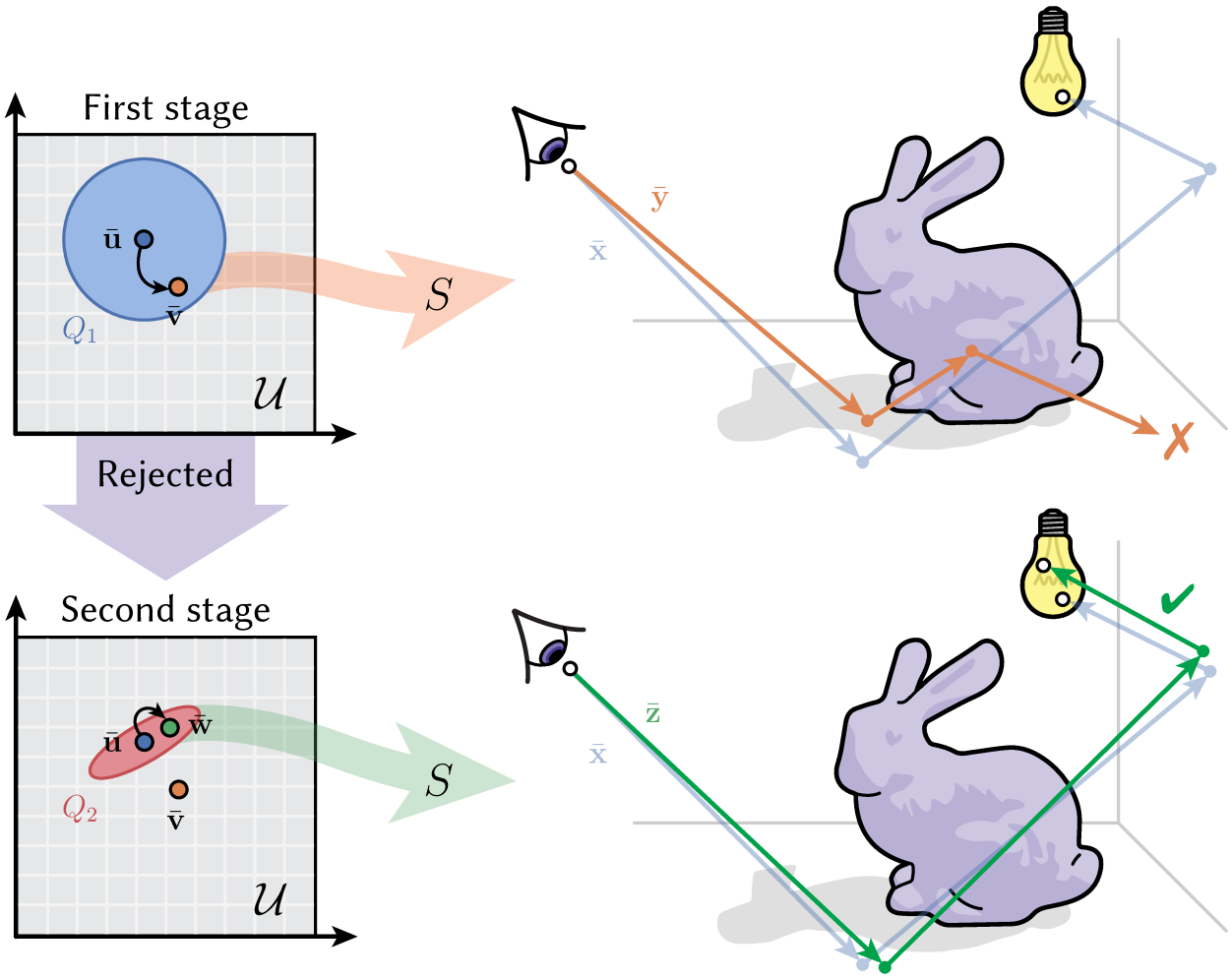


Figure 4.2: Illustration of delayed rejection-based algorithm. Mutating the primary sample $\bar{\mathbf{u}}$ using a first candidate transition kernel Q_1 yields a point $\bar{\mathbf{v}}$ that maps to light path $\bar{\mathbf{y}} = S(\bar{\mathbf{v}})$ in orange. In this example, the first stage proposal is rejected due to a significant (and unpredictable) change in the structure of the light path. A new vector $\bar{\mathbf{w}}$ is then proposed from $\bar{\mathbf{u}}$ using a second Markov kernel Q_2, producing path $\bar{\mathbf{z}}$ (in green) that is more likely to be accepted.

rates down. This solution is general and straightforward to implement atop *any* primary sample space (PSS) method.

We demonstrate the benefits of our method by integrating it into three algorithms: primary sample space Metropolis light transport (PSSMLT) [89], multiplexed Metropolis light transport (MMLT) [67] and Hessian Hamiltonian Monte Carlo (H2MC) [101]

(Figure 4.1). For a refresher on these topics, please refer to Section 3.2.5. Here, our two-stage variants outperform each of their original one-stage counterparts, resulting in smoother results on a set of challenging scenes with different lighting, geometry and material configurations.

Concretely, we present the following contributions:

- a two-stage proposal mechanism that adjusts to local structures in target densities,
- a novel candidate transition kernel that alleviates vanishing acceptance in the original DR method, and
- a benchmark of the versatility of two-stage kernels on several MCMC applications in rendering, improving convergence at equal time and with minimal implementation effort.

4.2 Two-Stage Delayed Rejection

We propose an extension of MLT-based algorithms to address the mutation strategy-selection problem. Our variant of DR [188, 58], tailored to the form of integrands we face in light transport, allows us to *conservatively* apply a sequence of strategies, i.e., only when necessary. At its core, DR modifies MH to admit different types of transitions at different stages. Suppose a first candidate mutation is generated and then rejected by MH. Rejection suggests that the proposal is not well suited and should be altered. Instead of retaining the current state and proceeding to the next iteration, a *different* candidate transition kernel can be used to propose a new state within the *same* iteration. This subtlety here is crucial as the application of state-dependent kernels would invalidate reversibility [2]. Under the DR formalism, the newly introduced kernel is allowed to depend on the previously rejected sample, and acceptance probabilities are computed in a manner that preserves reversibility.

This approach allows for the design of more versatile kernels that can be combined to propose transitions according to the underlying complexity of each mutation scenario,

leading to broader exploration of regions in state space at scales that fit them best. Our two-stage framework readily applies to several MCMC applications (Section 4.3), which we efficiently combine atop three existing PSS algorithms. In general, DR variants of these methods improve local exploration for difficult light paths without sacrificing performance in regions where simpler strategies suffice.

4.2.1 Delayed Rejection Metropolis–Hastings

We formally introduce the mathematical framework of DR [188]. Suppose we wish to draw samples from a target distribution π with density $\pi(\boldsymbol{x})$ over an arbitrary state space $S \subseteq \mathbb{R}^d$. A ***two-stage DR*** algorithm starts with a standard MH step—that is, proposing a first state $\boldsymbol{y}$ from $\boldsymbol{x}$ with density $Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})$ and accepting it with probability

$$\alpha_1(\boldsymbol{x}, \boldsymbol{y}) = 1 \wedge \frac{\pi(\boldsymbol{y})\, Q_1(\boldsymbol{x} \,|\, \boldsymbol{y})}{\pi(\boldsymbol{x})\, Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})}\,, \tag{4.1}$$

where the subscript on acceptance probability indicates the stage. If this first proposal is rejected, DR generates a new candidate $\boldsymbol{z}$ sampled from a different density $Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x})$, accepting it with probability

$$\alpha_2(\boldsymbol{x}, \boldsymbol{z}) = 1 \wedge \frac{\pi(\boldsymbol{z})\, Q_1(\boldsymbol{y} \,|\, \boldsymbol{z})\, Q_2(\boldsymbol{x} \,|\, \boldsymbol{y}, \boldsymbol{z})\, [1 - \alpha_1(\boldsymbol{z}, \boldsymbol{y})]}{\pi(\boldsymbol{x})\, Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})\, Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x})\, [1 - \alpha_1(\boldsymbol{x}, \boldsymbol{y})]}\,. \tag{4.2}$$

Here, we read conditionals *right-to-left*: $Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x})$ is the density of sampling $\boldsymbol{z}$ conditioned on first proposing $\boldsymbol{y}$ from the current state $\boldsymbol{x}$. The acceptance probability in Equation (4.2) greedily imposes detailed balance at both stages: the probability of proposing and rejecting a first candidate $\boldsymbol{y}$ is $Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})\, [1 - \alpha_1(\boldsymbol{x}, \boldsymbol{y})]$, and so the probability of moving to the second stage is $Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x}) Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})\, [1 - \alpha_1(\boldsymbol{x}, \boldsymbol{y})]$. This formulation is sufficient for preserving the chain's reversibility. When Q_2 is symmetric

with respect to $\boldsymbol{x}$ and $\boldsymbol{z}$ (i.e., $Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x}) \equiv Q_2(\boldsymbol{x} \,|\, \boldsymbol{y}, \boldsymbol{z})$), Equation (4.2) simplifies to

$$\alpha_2(\boldsymbol{x}, \boldsymbol{z}) = 1 \wedge \frac{\pi(\boldsymbol{z})\, Q_1(\boldsymbol{y} \,|\, \boldsymbol{z})\, [1 - \alpha_1(\boldsymbol{z}, \boldsymbol{y})]}{\pi(\boldsymbol{x})\, Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})\, [1 - \alpha_1(\boldsymbol{x}, \boldsymbol{y})]}\,. \tag{4.3}$$

Note that symmetry in Q_1 is not sufficient to eliminate it from the expression since, generally speaking, $Q_1(\boldsymbol{y} \,|\, \boldsymbol{z}) \neq Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})$.

4.2.2 Illustrative 1D Example

We demonstrate the behavior and benefits of DR on a simple example by Green and Mira [58]. Consider a one-dimensional target distribution π whose density is prescribed by a mixture of two uniform distributions:

$$\pi(x) = \frac{1}{2\alpha}\, \mathbf{1}_{[-\alpha, 0)}(x) + \frac{1}{2\beta}\, \mathbf{1}_{[0, \beta)}(x)\,, \tag{4.4}$$

where $\mathbf{1}_{[a,b]}$ is the indicator function on the interval $[a, b]$. Suppose $\alpha \ll \beta$, then when $x \in [-\alpha, 0)$ the candidate transition kernel should prefer smaller transitions, otherwise proposals are likely to be rejected as they often fall outside the support of π; however, when $x \in [0, \beta)$ larger transition kernels are preferable, otherwise the chain will mix slowly. In this case, the acceptance rate will be high but global exploration will be poor.

One option to address this scenario is to hand-tune the effective spread region of the candidate transition kernel to balance local exploitation and global exploration. Unfortunately, this tuning becomes difficult when the target density is complicated and irregular, which is almost always the case in light transport simulations. Alternatively, we can seek such a compromise directly through DR.

Figure 4.3 illustrates trace plots for a Markov chain whose limiting distribution is Equation (4.4) evolving with different transition mechanisms, both with the standard MH accept/reject step and the DR technique. Each plot graphs the current state of each

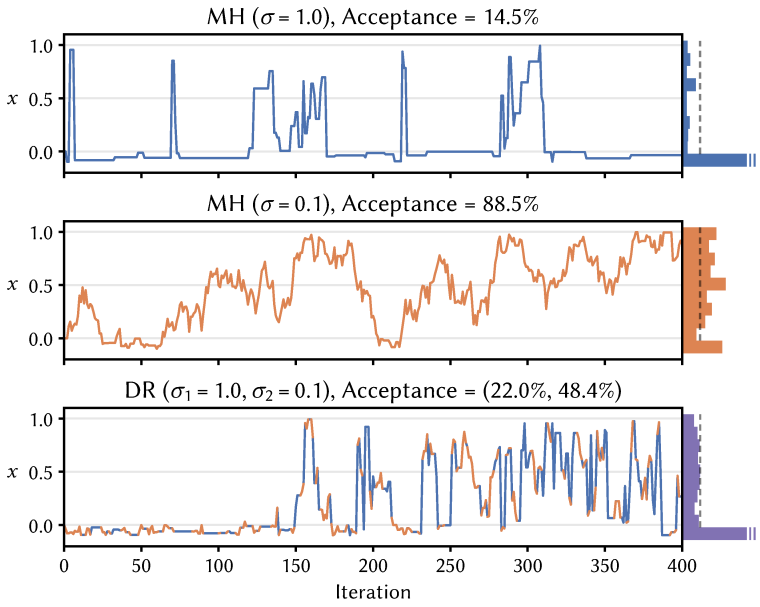


Figure 4.3: 1D trace plots of the state trajectory for MH and DR. Some bars exceed the display range; the vertical dashed line represents the positive side of the target $\pi|_{[0,1)}$. Top: A large proposal leads to long plateau of repeated states near zero and low acceptance. Middle: When the proposal is too small, the chain moves slowly from one region to another, resulting in high correlation between successive states as shown by mountain-like patterns. This results in under-exploration of the high probability region $[-1/10, 0)$. Bottom: DR provides a more expressive proposal mechanism that captures both parts of the target density. Colors correspond to which stage was accepted.

chain over time. We also plot the resulting histogram of the estimated distribution for each approach. Intuitively, $\pi(\boldsymbol{x})$ exhibits a large spike for $[-\alpha, 0)$ followed by a low plateau for $[0, \beta)$. This example uses $\alpha = nicefrac110$, $\beta = 1$ and a Gaussian proposal with variance σ^2. As expected, the chain gets stuck for larger σ and exhibits high autocorrelation for smaller σ. DR mitigates the drawbacks of both proposals and yields better exploration, as evidenced in its resulting histograms whose bins should match the dashed line over $[0, \beta)$.

Discussion: In this 1D example, the optimal spread of the kernel lies somewhere between α and β and hand-tuning σ would therefore not be too tedious: indeed,

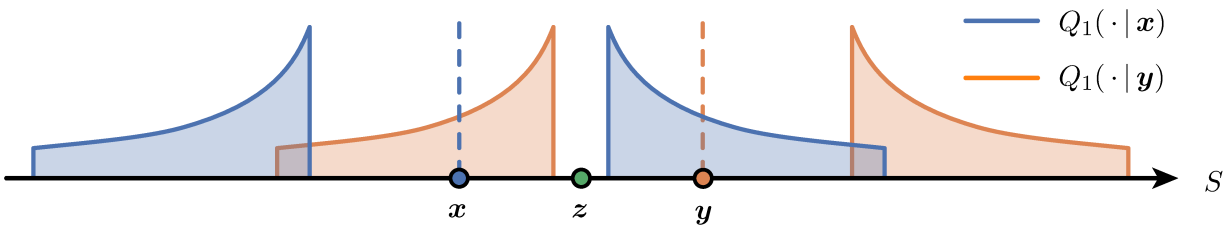


Figure 4.4: Failure of Tierney and Mira [188]. Given a first stage kernel Q_1 without global support over S (e.g., Kelemen-style mutation), it is possible to sample $\boldsymbol{z}$ in a zero-density region at the second stage, zeroing out $\alpha_2(\boldsymbol{x}, \boldsymbol{z})$. We refer to this problem as the *vanishing acceptance* at the second stage.

adaptive MCMC [63] could be employed to automatically tune σ. This scale, however, is global and might not be optimal for all regions. Relying on DR allows us to use multiple transition kernels and automatically revert to the one that is most suitable locally, improving chain mixing over (much) more of the domain.

4.2.3 Limitations of Original Framework

In general, Tierney and Mira's [188] original formulation in Section 4.2.1 can be quite restrictive. The simplicity of Equation (4.2) hides one important issue when computing the reverse probabilities: it implies that the backward path from $\boldsymbol{z}$ to $\boldsymbol{x}$ has to follow the forward path from $\boldsymbol{x}$ to $\boldsymbol{z}$ *with time reversed*. In other words, it has to *pass through* the intermediate state $\boldsymbol{y}$ and get rejected. This can be problematic when $\boldsymbol{y}$ is far from $\boldsymbol{z}$, in which case the term $Q_1(\boldsymbol{y} \mid \boldsymbol{z}) \to 0$ as does $\alpha_2(\boldsymbol{x}, \boldsymbol{z})$. This holds even when Q_1 is symmetric since it is not simplified by cancellation from a term in the denominator. A similar situation occurs when Q_1 does not have support near its center and $\boldsymbol{y}$ is too close to $\boldsymbol{z}$ (Figure 4.4). This ***vanishing acceptance*** behavior is more pronounced when the candidate transition kernels used at both stages have limited overlapping densities. This configuration increases the likelihood of sampling a second point in the tails of the first proposal density, which may occur frequently when dealing with narrow, i.e., exponential distributions. The same issue was indirectly observed by Green and Mira

[58] and later rediscovered by Trias et al. [189]. To the best of our knowledge, we are the first to confront this issue directly within the DR formalism.

To see why doing this limitation affects light transport simulation, consider the exponential candidate transition kernel recommended by Kelemen et al. [89]. This so-called ***Kelemen-style*** mutation strategy proposes new states $\boldsymbol{y} = (y_1, \ldots, y_d)$ from $\boldsymbol{x}$ as

$$y_i = x_i + \operatorname{sgn}\left(\xi_1 - \tfrac{1}{2}\right)\epsilon_{\mathsf{max}} \exp\left(-\log\frac{\epsilon_{\mathsf{max}}}{\epsilon_{\mathsf{min}}}\xi_2\right), \tag{4.5}$$

where $\xi_{1,2} \sim U[0,1]$ are uniform variates and $0 < \epsilon_{\mathsf{min}} < \epsilon_{\mathsf{max}}$ are parameters controlling the kernel size with $|y_i - x_i| \in [\epsilon_{\mathsf{min}}, \epsilon_{\mathsf{max}}]$. Kelemen et al. [89] originally suggested setting $(\epsilon_{\mathsf{min}}, \epsilon_{\mathsf{max}}) = (1/1024, 1/64)$. This kernel has zero support around zero, forcing minimal-distance moves from the current state, which encourages image plane stratification [183] by driving newly proposed paths towards different pixel locations. The resulting density is symmetric and is the product of k independent probability density function (PDF) at each primary sample, allowing for lazy path construction. One consequence of this zero-probability "hole" is that we cannot directly apply DR when first stage samples are drawn from a Kelemen-style kernel, as the mutation does not have global support. We propose two approaches to address this *vanishing acceptance* (Figure 4.4), each of which motivates a different application to light transport simulations (Section 4.3).

4.2.4 Pairwise Orbital Mutations

We first propose a novel perturbation technique designed to remove the need for computing the ratio $\Gamma_1 \triangleq Q_1(\boldsymbol{y} \,|\, \boldsymbol{z}) \,/\, Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})$ in Equation (4.3). Suppose that Q_1 is a product of circularly symmetric (i.e., $Q_1(\boldsymbol{y} \,|\, \boldsymbol{z}) \equiv Q_1(\|\boldsymbol{y} - \boldsymbol{z}\|)$), independent and lower-dimensional densities that partition the state space. Our key insight is that sampling $\boldsymbol{z} \sim Q_2(\boldsymbol{z} \,|\, \boldsymbol{y})$ such that $\|\boldsymbol{z} - \boldsymbol{y}\| = \|\boldsymbol{y} - \boldsymbol{x}\|$ for each element of the partition is sufficient for the ratio Γ_1 to cancel out. Since surface light transport with PSS typically

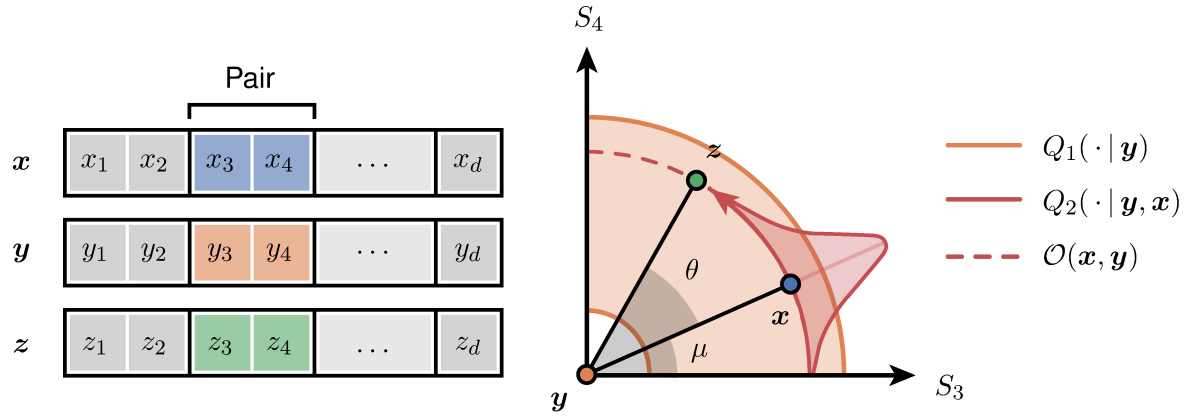


Figure 4.5: Our pairwise orbital mutation. When a first state $\boldsymbol{y}$ is rejected, a second state $\boldsymbol{z}$ is greedily sampled from a Wrapped Cauchy distribution Q_2 such that it lies on the orbit $\mathcal{O}(\boldsymbol{x},\boldsymbol{y})$ for each pair of primary samples. This removes the need to account for transition ratios in the acceptance ratio.

uses pairwise vertex relationships, we group coordinates pairwise and sequentially, e.g., (y_1, y_2), (y_3, y_4), and so on. This corresponds to perturbing directional samples at each vertex. Visually, such a second stage proposal amounts to moving along a circle that is centered at $\boldsymbol{y}$ and that passes through $\boldsymbol{x}$ (Figure 4.5).

Any $\boldsymbol{z}$ located on the orbit $\mathcal{O}(\boldsymbol{x},\boldsymbol{y}) \triangleq \{\boldsymbol{o} : \|\boldsymbol{o}-\boldsymbol{x}\| = \|\boldsymbol{y}-\boldsymbol{x}\|\}$ will satisfy our needs. Ideally, we want $\boldsymbol{z}$ to be close to $\boldsymbol{x}$ on the orbit $\mathcal{O}$ to increase the likelihood of acceptance. We denote the direction between the current and the proposed state as $\boldsymbol{w} = \boldsymbol{x} - \boldsymbol{y}$, the radius of the orbit as $r = \|\boldsymbol{w}\|$ and the angle between this direction and the first coordinate axis as μ. In order to sample a point close to $\boldsymbol{x}$ on $\mathcal{O}$, we wish to sample a small azimuthal θ such that the angle between $\boldsymbol{z}-\boldsymbol{y}$ and the first coordinate axis is $\mu' = \mu + \theta$.

We construct circular distributions [48], over a domain with interval $[0, 2\pi)$, by warping 1D PDFs p periodically over the unit circle. We choose the ***wrapped Cauchy distribution***, with PDF

$$Q_{\text{WCauchy}}(\theta;\rho) = \frac{1}{2\pi}\frac{1-\rho^2}{1+\rho^2-2\rho\cos\theta}\,, \tag{4.6}$$

where the scale $\rho \in [0, 1]$ plays a role analogous to the standard deviation σ in a 1D normal distribution. This specific choice is motivated by the distribution admitting a closed-form expression for its cumulative distribution function—a requirement for inversion sampling. The distribution approaches a uniform circular density as $\rho \to 0$, whereas when $\rho \to 1$ the distribution degenerates into a Dirac comb (namely, a 2π-periodic tempered Dirac). We finally recover the second stage sample components by projecting back onto the axis $\boldsymbol{z} = \boldsymbol{y} + r\,(\cos\mu', \sin\mu') \in \mathcal{O}$. Note that this is similar in spirit to elliptical slice sampling [135], except we do not have to iterate the sampler to obtain a new point on the orbit.

By construction, the resulting kernel satisfies the $Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x}) \equiv Q_2(\boldsymbol{x} \,|\, \boldsymbol{y}, \boldsymbol{z})$ condition and the new state $\boldsymbol{z}$, lying on $\mathcal{O}$, implies that $Q_1(\boldsymbol{y} \,|\, \boldsymbol{z}) \equiv Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})$. This results in the simplified acceptance ratio

$$\alpha_2(\boldsymbol{x}, \boldsymbol{z}) = 1 \wedge \frac{\pi(\boldsymbol{z})\,[1 - \alpha_1(\boldsymbol{z}, \boldsymbol{y})]}{\pi(\boldsymbol{x})\,[1 - \alpha_1(\boldsymbol{x}, \boldsymbol{y})]} = 1 \wedge \frac{0 \vee [\pi(\boldsymbol{z}) - \pi(\boldsymbol{y})]}{\pi(\boldsymbol{x}) - \pi(\boldsymbol{y})}\,, \tag{4.7}$$

where $0 \vee \eta \triangleq \max(0, \eta)$. This acceptance probability can be evaluated efficiently in a Metropolis sampler. We use these ***orbital mutations*** and this ratio in our experiments when the candidate transition kernel of the baseline MCMC sampler factors into a product of independent components.

It is worth noting that this particular combination of pairwise mutations and the wrapped Cauchy distribution is not the only approach one could use to obtain a simplified acceptance ratio. When Russian roulette is used in PSS or in the presence of participating media, correlating triplets of samples (e.g., to account for free-flight distance sampling) would be better suited than pairs. This could be achieved, for instance, by sampling directions on a sphere according to a von Mises–Fisher distribution [69].

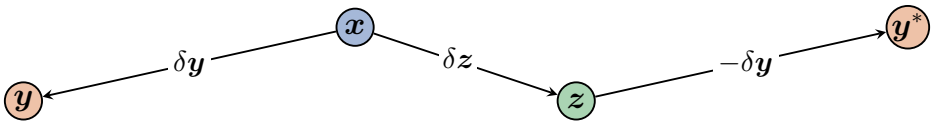


Figure 4.6: Generalized DR. Different states involved in Green's two-stage DR algorithm using a fictional state $\boldsymbol{y}^* = \boldsymbol{z} - \delta\boldsymbol{y}$. Even though $Q_1(\boldsymbol{y} \,|\, \boldsymbol{z}) \neq Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})$ in general, we do have that $Q_1(\boldsymbol{y}^* \,|\, \boldsymbol{z}) = Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})$ for a symmetric Q_1.

Limitations: Generalizing our construction to higher dimensions with hyper-spherical distributions is feasible but more involved. Such a generalization would also require correlated multi-dimensional samples, and so would induce significant changes to underlying path sampling routines. We chose the pairwise approach for its simplicity and ease of integration atop traditional PSSMLT and lazy path evaluation, leaving these alternatives to future work.

While our orbital mutation can solve the vanishing acceptance problem, it reduces the second proposal's choice to the choice of a circular PDF. In fact, restricting the second state to lie on an orbit prevents our current method from using correlated proposals, such as the anisotropic Gaussian mutations proposed by Li et al. [101]. Using such mutation strategies without the orbital constraints would reintroduce the vanishing acceptance problem as was discussed in Section 4.2.3. To remedy to this limitation, we propose an extension of our approach based on the work of Green and Mira [58].

4.2.5 Generalized Delayed Rejection

With reversible jump Markov chain Monte Carlo (RJMCMC), Green and Mira [58] relax the $\boldsymbol{x} \leftrightarrow \boldsymbol{y} \leftrightarrow \boldsymbol{z}$ reversibility constraint by employing an additional fictional state $\boldsymbol{y}^*$ that effectively generalizes the original DR algorithm to transdimensional moves. Below, we restrict ourselves to fixed dimensional mappings and present only one case of specific interest. For the full theoretical exposition we refer to Green and Mira [58].

Similar to the original formulation, the first move of the ***generalized two-state DR framework*** consists of generating a sample $\boldsymbol{y} = \boldsymbol{x} + \delta\boldsymbol{y}$ in a standard MH step. If the move is rejected, we generate a new state $\boldsymbol{z} = \boldsymbol{x} + \delta\boldsymbol{z}$ sampled according to a different density $Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x})$ and then consider an ***intermediate, fictional state*** $\boldsymbol{y}^* = \boldsymbol{z} - (\boldsymbol{y} - \boldsymbol{x}) = \boldsymbol{z} - \delta\boldsymbol{y}$ instead of $\boldsymbol{y}$ (Figure 4.6). The new state $\boldsymbol{z}$ is then accepted with probability

$$\alpha_2^*(\boldsymbol{x}, \boldsymbol{z}) = 1 \wedge \frac{\pi(\boldsymbol{z})\, Q_1(\boldsymbol{y}^* \,|\, \boldsymbol{z})\, Q_2(\boldsymbol{x} \,|\, \boldsymbol{y}^*, \boldsymbol{z})\, [1 - \alpha_1(\boldsymbol{z}, \boldsymbol{y}^*)]}{\pi(\boldsymbol{x})\, Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})\, Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x})\, [1 - \alpha_1(\boldsymbol{x}, \boldsymbol{y})]}\,, \tag{4.8}$$

where

$$\alpha_1(\boldsymbol{z}, \boldsymbol{y}^*) = 1 \wedge \frac{\pi(\boldsymbol{y}^*)\, Q_1(\boldsymbol{z} \,|\, \boldsymbol{y}^*)}{\pi(\boldsymbol{z})\, Q_1(\boldsymbol{y}^* \,|\, \boldsymbol{z})}\,. \tag{4.9}$$

Note that the presence of $\boldsymbol{y}^*$ in Equation (4.8) induces symmetry. Contrary to the initial formulation, we observe that its contribution in every stage disappears when Q_1 is symmetric:

$$\alpha_2^*(\boldsymbol{x}, \boldsymbol{z}) = 1 \wedge \frac{\pi(\boldsymbol{z})\, Q_2(\boldsymbol{x} \,|\, \boldsymbol{y}^*, \boldsymbol{z})\, [1 - \alpha_1(\boldsymbol{z}, \boldsymbol{y}^*)]}{\pi(\boldsymbol{x})\, Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x})\, [1 - \alpha_1(\boldsymbol{x}, \boldsymbol{y})]}\,, \tag{4.10}$$

solving the acceptance problem we discussed earlier. The special case $\boldsymbol{y}^* = \boldsymbol{y}$ is precisely Equation (4.2) from Tierney and Mira [188]. Green and Mira [58] provide the full derivation of Equation (4.10).

While Green and Mira's [58] approach generally gives comparable acceptance to our orbital mutation—and thus higher than Tierney and Mira's [188]—it involves an extra evaluation of the target density at $\boldsymbol{y}^*$. In the context of light transport simulation, this requires tracing an additional light path. As shown in Figure 4.7, this overhead often results in lower performance than our orbital mutation when used in simpler applications. In such cases, the orbital approach should be preferred. However, when

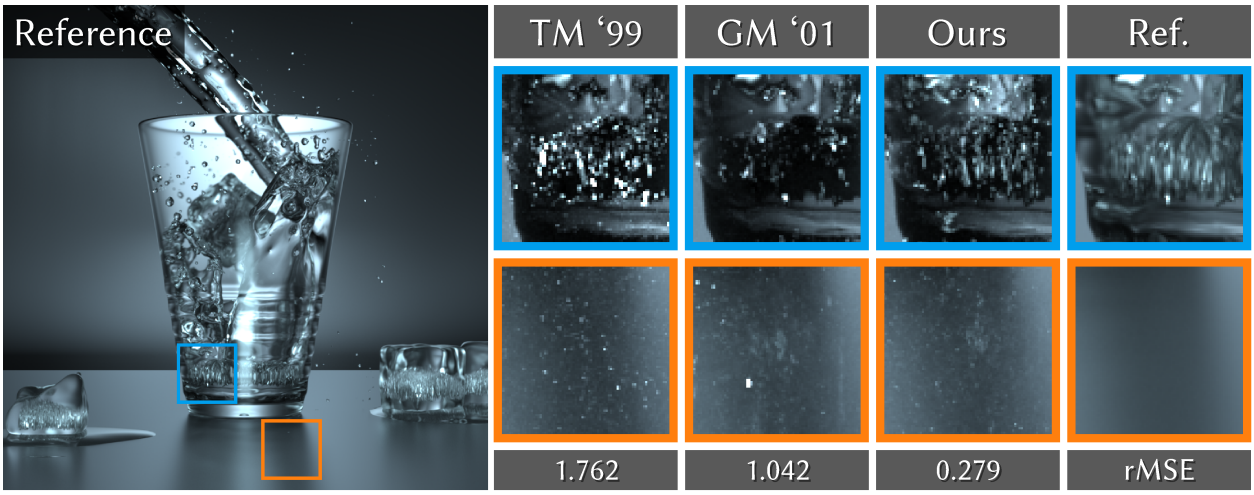


Figure 4.7: Comparison of orbital mutations. Equal time comparison of Tierney and Mira [188], Green and Mira [58] and our pairwise orbital approach on the GLASS OF WATER scene.

one wishes to use mutation strategies that are not compatible with orbital mutations (such as H2MC [101]), Green and Mira's [58] generalization should be used.

4.2.6 Waste-recycling for Proposed States

By design, DR generates more states than a standard MH step. While accumulating only accepted states is viable, we want a scheme that involves *all* states to avoid dismissing important information. To this end, we adapt the ***use of the expected values*** [193] to our two-stage proposal mechanism.

Consider the following observations:

- moving from state $\boldsymbol{x}$ to the first proposal $\boldsymbol{y}$ has probability $\omega_{\boldsymbol{y}} = \alpha_1(\boldsymbol{x}, \boldsymbol{y})$,
- getting rejected at the first stage has probability $1 - \alpha_1(\boldsymbol{x}, \boldsymbol{y})$,
- once at $\boldsymbol{y}$, moving to a second proposal $\boldsymbol{z}$ has probability $\alpha_2(\boldsymbol{y}, \boldsymbol{z})$,
- the probability of the full sequence $\boldsymbol{x} \to \boldsymbol{y} \to \boldsymbol{z}$ occurring and getting accepted at $\boldsymbol{z}$ is $\omega_{\boldsymbol{z}} = (1 - \alpha_1)\alpha_2$, and
- similarly, the probability of both proposed stages failing and thus staying at $\boldsymbol{x}$ is $\omega_{\boldsymbol{x}} = 1 - \omega_{\boldsymbol{y}} - \omega_{\boldsymbol{z}} = (1 - \alpha_1)(1 - \alpha_2)$.

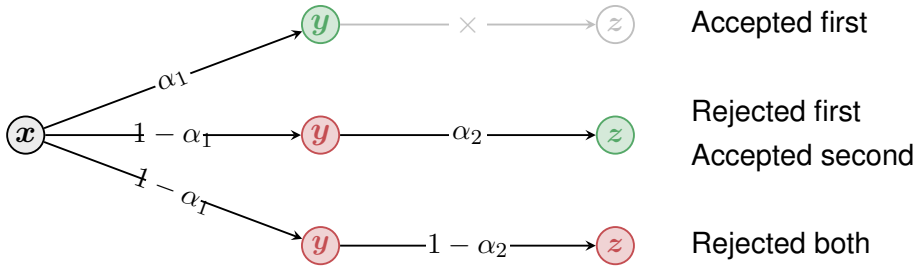


Figure 4.8: Decision tree. Decision tree of our DR algorithm.

We accumulate sample contributions at all three locations, weighted by their corresponding probabilities $\omega_\square$ (Figure 4.8). We observe that, when using Equation (4.8), since $\boldsymbol{y}^*$ is not a state that is directly accessible from $\boldsymbol{x}$, we cannot splat its contribution using the method of expected values. Indeed, this technique amounts to accumulating $\mathbb{E}\left[\pi(X_i) \,|\, X_{i-1} = \boldsymbol{x}\right]$, but $\boldsymbol{y}^*$ was sampled from $\boldsymbol{z}$, not from $\boldsymbol{x}$.

4.2.7 Discussion on Mixture Models

While mixture models can simulate different candidate transition kernels, they remain agnostic to the current state of the chain and rely on careful mixture weight settings. In contrast, delaying the rejection of samples provides an opportunity to adjust proposals along the chain, automatically tempering the negative consequences of local mismatches between the candidate transition kernels and target density. Our DR approach applies to MCMC light transport in order to afford chains a "second chance" to reach valuable regions in path space.

4.3 Applications and Results

We validate our theory by employing our two-stage kernel to augment various MCMC light transport algorithms with more flexible transitions. We demonstrate that this simple—yet effective—addition increases the robustness of several techniques operating in PSS. We group our applications into two categories:

(I) **Bold-then-Timid (Section 4.3.1).** The first stage attempts a more adventurous transition; if it fails, we propose a more conservative move. Larger perturbations can improve exploration but are more likely to be rejected; smaller perturbations provide a "safety net" to avoid repeating the same state.

(II) **Cheap-then-Expensive (Section 4.3.2).** The first stage uses a simple, efficient mutation; if it fails, a more intricate kernel is employed. This amortizes the cost of specialized mutations, limiting their use to challenging regions in state space.

We show the pseudocode of our algorithm in Appendix B. We analyze the performance of our algorithm by tracking the relative mean squared error (rMSE) compared to a reference image, reporting the median over several runs. We focus on this error metric as it is more sensitive to fireflies, i.e. error artifacts caused by poor exploration. All reference images are generated with several days of render time in their respective baseline renderers. All comparisons are equal time renderings running on an Intel Platinum 8160F Skylake CPU at 2.1 GHz with 48 threads.

4.3.1 Bold-then-Timid

First, we present our bold-then-timid application results. Note that we present two different applications of this framework, one combining Kelemen then pairwise orbital mutations, and one combining multiplexed then pairwise orbital subpath mutations.

4.3.1.1 Kelemen then Pairwise Orbital

The Kelemen kernel is the de facto mutation choice for most PSS-based methods. It can, however, be detrimental to the chain when the density is concentrated in thin regions of the space. In these scenarios, off-centered perturbations can lead to poor local exploration of thin highlights, which would be better served by a more localized candidate transition kernel. As these small regions of the state space contribute significant energy to the final image, locally-bad fits introduce a disproportional amount

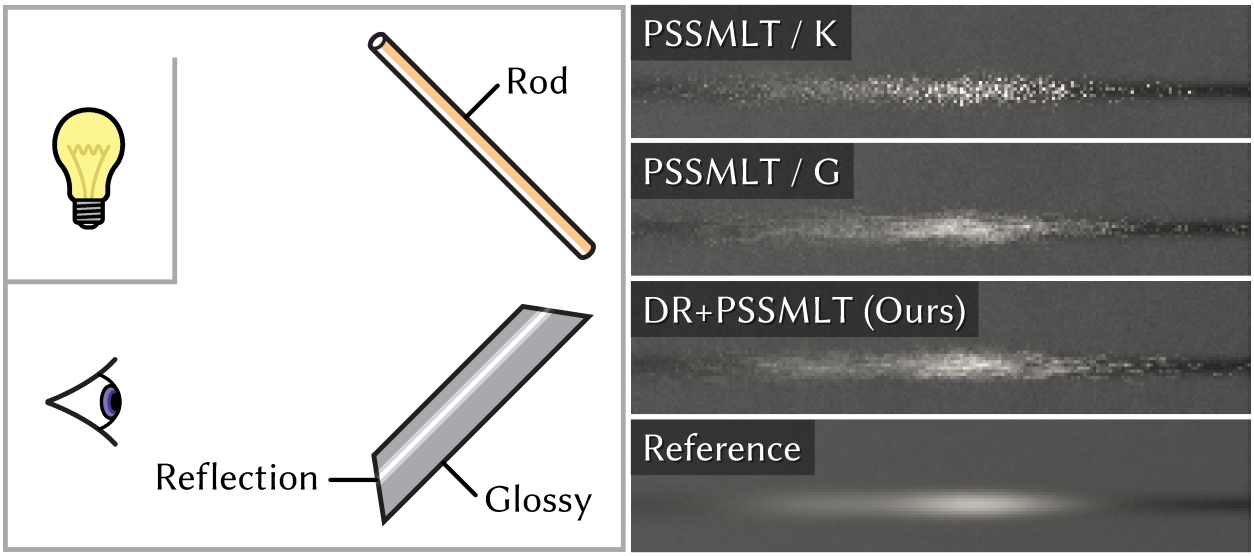


Figure 4.9: Failure of Kelemen-style mutation. This scene is explicitly designed to fail bold mutations as they cannot efficiently explore the thin highlight. Here, the Gaussian kernel (PSSMLT / G) clearly wins over Kelemen (PSSMLT / K). Even if our two-stage mechanism starts with a Kelemen mutation, it is able to recover and efficiently explore the reflection on the plate.

of noise in these regions (Figure 4.9). Identifying occurrences that lead to these problems directly within a Metropolis sampler is thus difficult.

We use our pairwise orbital perturbation (Section 4.2.4) to reach a good compromise by mutating paths with a more *timid* candidate transition kernel. In resorting to this more conservative proposal, chains can suggest paths that are closer to the current one—while maintaining a sufficiently high acceptance rate—without staying at the same state for many iterations. This extra stage has the desired effect of slightly moving the current path and provides an effective mechanism to better explore local modes.

Results: We integrated delayed rejection Metropolis light transport (DRMLT) atop a PSSMLT path tracer in Mitsuba v0.5 [77]. We use the kernel bounds recommended by Kelemen et al. [89] and our orbital perturbation with $\rho = \exp(-1/4)$ for the second stage. We keep this value fixed across all the experiments that use our orbital mutation. We

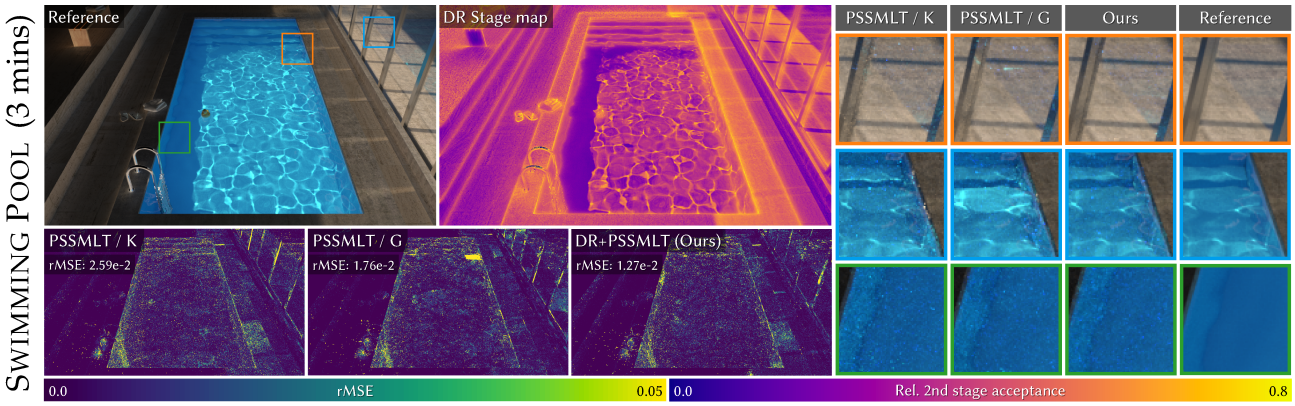


Figure 4.10: Comparison of bold-then-timid. Equal time comparison of the SWIMMING POOL scene for DR applied to PSSMLT. Compared to PSSMLT with the standard Kelemen-style kernel (PSSMLT / K) and with a Gaussian kernel (PSSMLT / G), DRMLT produces smoother results, thanks to our orbital mutation.

also show results with a Gaussian density with variance chosen to match the acceptance rate of our DR application. Doing so illustrates the versatility of our two-stage approach compared to a single-stage default kernel.

Figure 4.10 provides an equal time comparison on the SWIMMING POOL scene with complex transport from caustic and specular-diffuse-specular paths. Standard Kelemen-style mutations (PSSMLT / K) are too sensitive and have difficulties staying on the specular manifold, yielding noisy images. On the other hand, Gaussian mutations (PSSMLT / G) are less likely to fall outside a high density region, but they are too localized and generate luminosity spikes in the image. Our two-stage algorithm (DR+PSSMLT) works well overall, accommodating both types of mutations. Our method also produces a smoother output, thanks to the two-stage pixel splatting that makes use of all intermediate states. The DR stage map visualizes per-pixel relative acceptance at the second stage: as expected, a higher proportion of states are accepted at the second stage in complex regions, i.e., the caustics at the bottom of the pool and the reflected light around the pool.

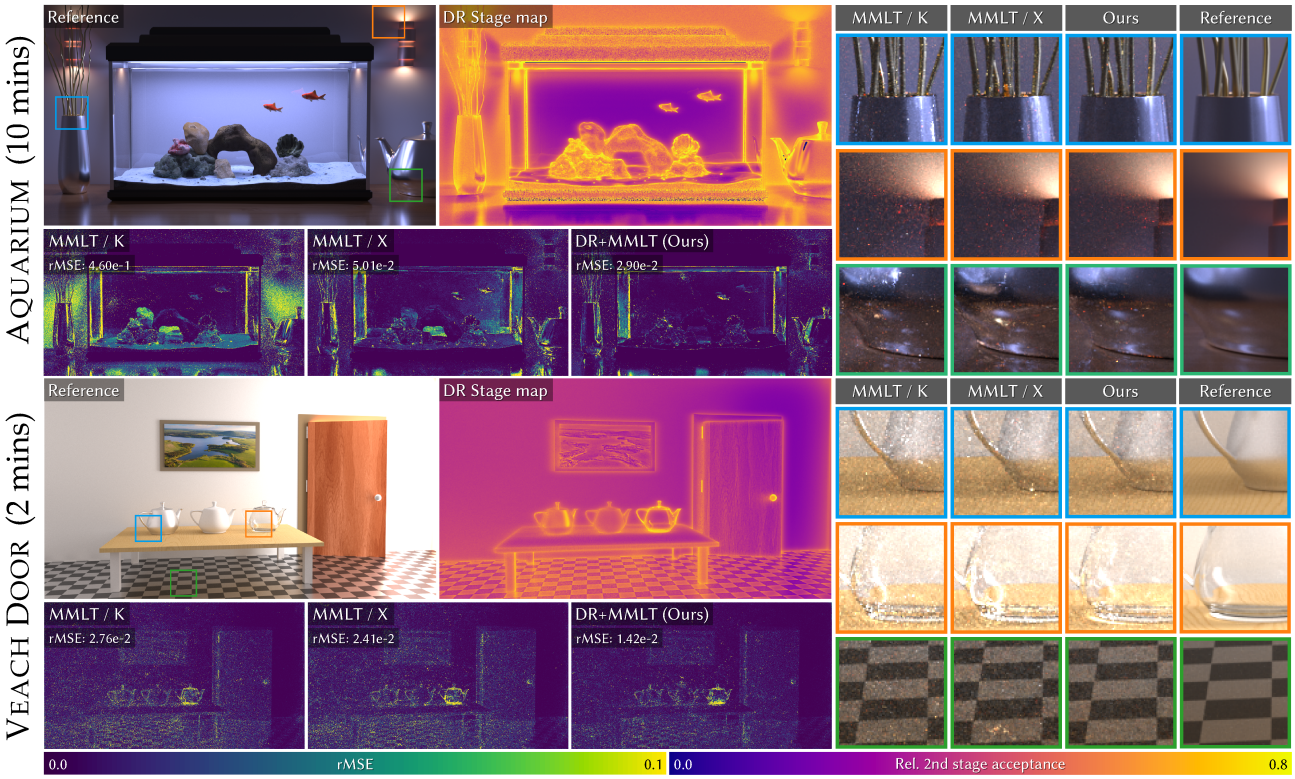


Figure 4.11: Comparison of multiplexed-then-subpath. Equal time comparisons of the scenes (AQUARIUM and VEACH DOOR) for DR applied to MMLT. By using a more flexible multiplexed kernel, our method significantly reduces variance compared to MMLT with a Kelemen-style kernel (MMLT / K) and a mixture model (MMLT / X).

4.3.1.2 Multiplexed then Subpath

Our bold-then-timid approach can further be applied to MMLT [67] to improve its efficiency. MMLT allows Markov chains to adaptively select the (s,t)-bidirectional sampling strategy, where s,t are the length of the emitter and sensor subpaths. Here, only one path is computed at each iteration, avoiding the computation of all-pairs MIS-weighted vertex contributions.

We only slightly modify MMLT when performing local perturbations, *fixing* the bidirectional strategy instead of allowing it to vary along the chain. Since strategy changes are rarely proposed and accepted [14], this modification does not harm performance as most of MMLT's gains come from evaluating a single path instead of an

entire family of paths. We perturb all primary samples with our *bold* mutation; for our *timid* mutation, we *fix* the emitter subpath and only perturb the sensor subpath samples using our pairwise orbital mutation. This choice allows us to reuse parts of the computations from current and rejected states as only the sensor subpath needs to be re-traced.

Results: We integrated MMLT atop Mitsuba v0.5. Figure 4.11 compares our DR+MMLT approach to standard MMLT on scenes with complex visibility. We also measure our performance against MMLT using an equally weighted mixture of our two stages (MMLT / X). Our DR+MMLT generates smoother images, has lower rMSE (on both scenes) and performs better than its mixture counterpart. This further supports our claim that, unlike mixture variants, our method is able to more appropriately (and automatically) select transition kernels based on the target density. AQUARIUM exhibits many light transport phenomena, evidenced in its stage acceptance map. The interior of the aquarium is lit by an emitter inside its casing, and is fairly easy to explore compared to the rest of the scene. In contrast, the first stage for VEACH DOOR behaves well except for a few localized regions. The constrast in the map—e.g., around the golden fishes and the wooden door—corresponds to abrupt changes in luminosity which tend to be initially rejected by our sampler.

Handling Light Tracing: Extra care is needed when a full emitter path forms the initial state of the chain, i.e., when $t = 1, s = k - 1$ (k is the path length). In this scenario, we mutate the entire path twice using our orbital mutation at the second stage to avoid proposing a state identical to the previous one. This case alone motivates our fixing of the bidirectional strategy across the chain: if a path gets mutated to a full emitter one, extra considerations would be needed to account for the sudden change of proposal distribution, without violating reversibility. Therefore, we opt for a more transparent

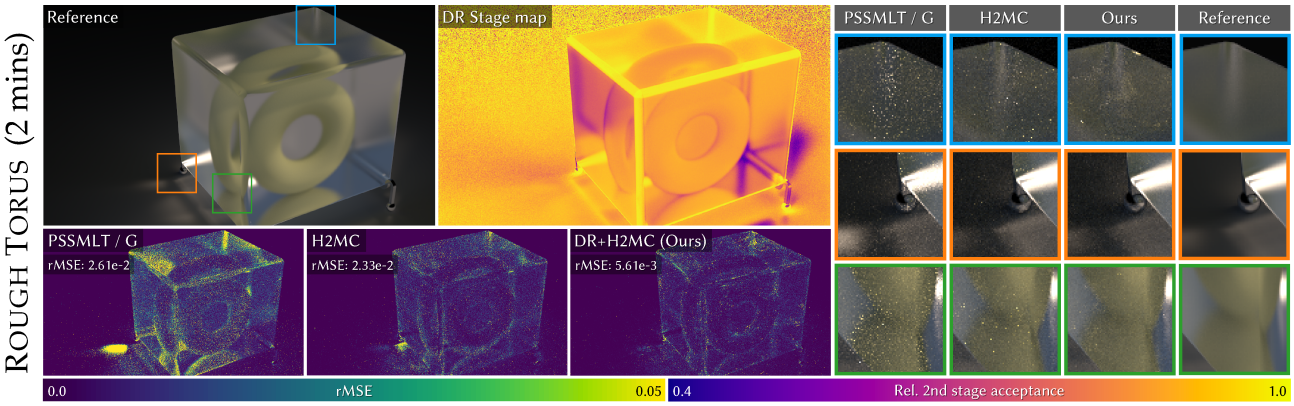


Figure 4.12: Comparison of cheap-then-expensive. Equal time comparison of the ROUGH TORUS scene for DR applied to H2MC. Our two-stage method explores difficult regions with strong directional transport and alleviates fireflies, compared to one-stage PSSMLT with an isotropic Gaussian kernel (PSSMLT / G) and fully anisotropic H2MC.

approach that treats pure light tracing as a special case without modifying the mathematical formulation of the acceptance ratios.

4.3.2 Cheap-then-Expensive

Simpler mutations, such as the one proposed by Kelemen, are less expensive and can be applied more often than complex ones (in fixed time). When working with limited resources and a target density that is nontrivial to evaluate, choosing when to apply a more expensive/complicated candidate transition kernel requires domain expertise. We propose to use of DR to automate this choice.

The H2MC proposal [101] uses automatic differentiation to compute first- and second-order derivatives for every possible pair of materials. This is costly (and so performed as a preprocess) but enables more expressive transitions. Under the H2MC framework, a d-dimensional state $\boldsymbol{y}$ is proposed from $\boldsymbol{x}$ as $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\gamma}$, where $\boldsymbol{\gamma} \sim \hat{n}(\boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\Sigma}_{\boldsymbol{x}})$ is sampled from a multivariate (correlated) normal density. Here, $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ is an $d \times d$ covariance matrix that depends on the throughput Hessian and $\boldsymbol{\mu}_{\boldsymbol{x}}$ is a mean variable; both depend on the *full* vector of random numbers $\boldsymbol{x}$. Most importantly, this correlation prevents the matrix

from factoring into a product of joint densities. As such, sampling from the H2MC kernel requires a costly eigendecomposition of the inverse Hessian.

To amortize this cost, Li et al. [101] propose to branch to anisotropic mutations based on a thresholded value of the Hessian L^2-norm, conservatively falling back to simple isotropic Gaussian transitions. The main issue with this approach is that it cannot capture anisotropy *across scales*, which limits exploration potential of the Hessian-based candidate transition kernel.

We leverage the generalized DR framework (Section 4.2.5) to determine this branching more efficiently. We always perform a cheaper, isotropic Gaussian transition first; if this fails, we resort to the more costly H2MC transition. Note that by doing so, we need only compute the extra Hessian for the second stage at $\boldsymbol{z}$. To demonstrate DRMLT's flexibility, we adopt a hybrid approach with a second stage transition set to an equally-weighted mixture of anisotropic and isotropic Gaussians, the latter with smaller variance. This amortizes the cost of computing $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ every time a first proposal is rejected.

Results: We demonstrate the robustness of our approach by implementing DR in the differential DPT renderer [101]. To ensure a fair comparison between DR+H2MC and H2MC, we disable path space lens perturbations in DPT and use their state space reparameterization for both techniques. We apply the same (original) parameters and a maximum path length of $k = 10$, allowing derivatives to be precomputed in a reasonable amount of time. We compare to PSSMLT with a Gaussian kernel matching our first stage's proposal and to H2MC (Figure 4.1 and Figure 4.12).

We chose to use PSSMLT with a Gaussian candidate transition kernel as an example of a cheap mutation strategy; this doubles as a baseline and the first stage in our DRMLT method. CHESS has many modes of transport, and so is challenging for an isotropic- or anisotropic-only proposal. ROUGH TORUS exhibits significant state space anisotropies well-suited to the Hessian-Hamiltonian dynamics-based proposals.

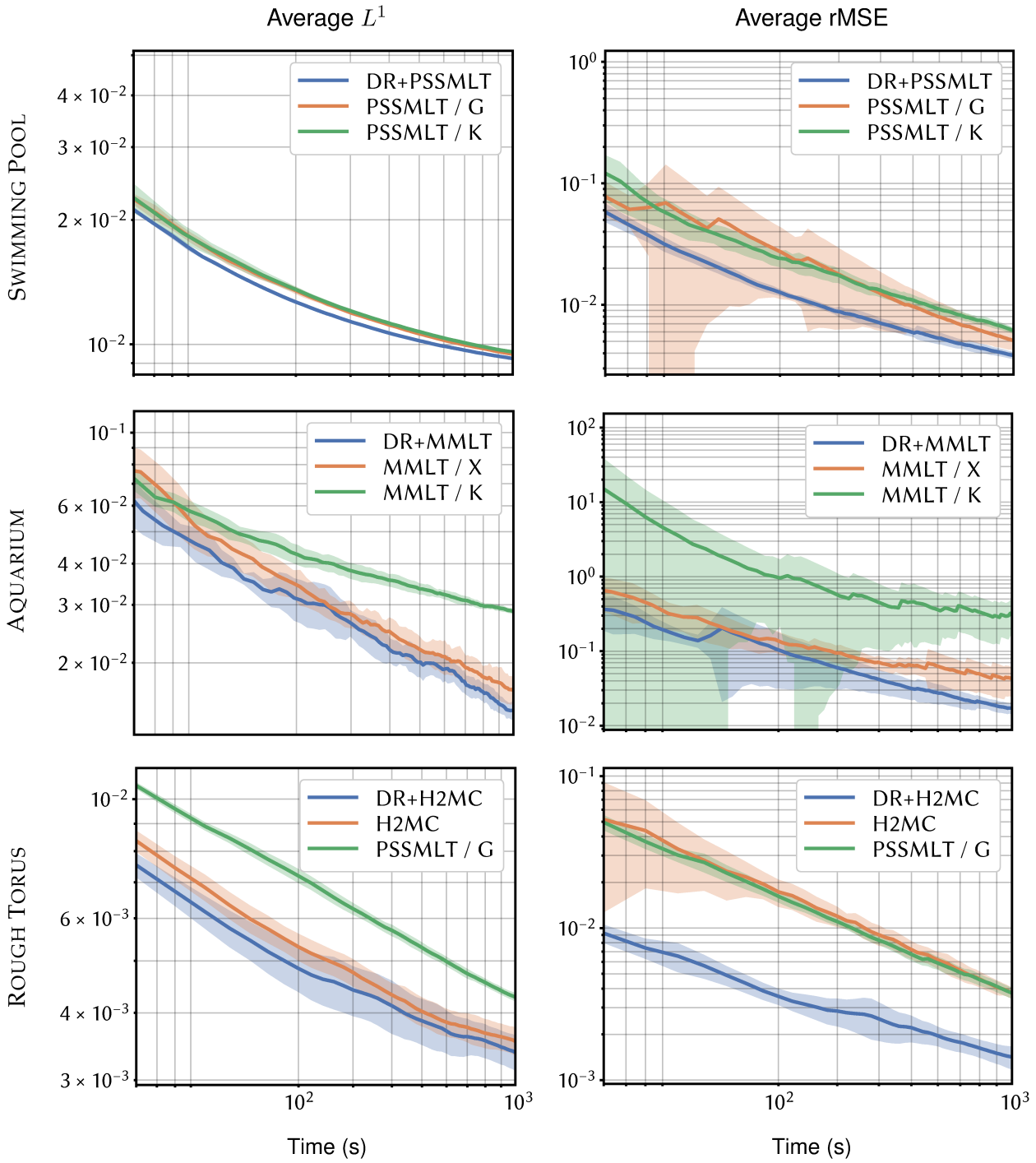


Figure 4.13: Convergence plots. We visualize both the mean (thick curve) and standard deviation (shaded regions) of the error over 10 independent runs. Plots for all scenes and other metrics can be found in our additional material.

Our method outperforms H2MC in both test scenes, as evidenced by lower errors and less variance in the rendered images. On CHESS, isotropic Gaussian proposals are more effective in regions where the target is smooth, but struggle with directional transport such as on rough glass. In contrast, H2MC fairs better on brushed metals and dielectrics, i.e., the chess pieces, but worse overall due to its overhead. ROUGH TORUS is dominated by transport concentrated in narrow bands of the space, as shown by the heavily skewed DR stage map, and our algorithm effectively balances the two stages and removes bright spikes in the image. Our method automatically finds an equilibrium between the two scenarios. Since first-stage mutations are an order of magnitude faster to evaluate, we are able to achieve more mutations in equal time, leading to consistent improvements across scenes.

4.3.3 Empirical Convergence Analysis

Due to correlation between MCMC samples (as opposed to independent samples), a single run of an MCMC integrator is not representative of its ability to explore the relevant space. Therefore, we run 10 instances of each algorithm initialized at different random states to better capture the average behavior. Since the normalization factor can vary from one initialization to another, we integrate the luminance of the reference images to compute global scaling factors and supply these values to every algorithm. Since this constant is common to every method compared against, doing so allow us to more efficiently compare the different algorithms. We also compute the standard deviation of the metrics and plot continuous error bars to visualize the variation, to assess the stability of our method.

Figure 4.13 shows one result per application. We refer to our supplementary material for more results. Our DRMLT approach **consistently outperforms** its one-stage PSS counterparts, both in L^1 and relative mean squared error (MSE). Our L^1-norm performance hints to a potential smoothing effect in low variance regions, due to our

double-splatting. In contrast, rMSE penalizes large outliers, such as fireflies. Our method is robust to both metrics, as shown by the lower curves.

Chapter 5

A Monte Carlo Method for Fluid Simulation (MCFluid)

We investigate the potential for enhancing the robustness and flexibility of fluid simulations when dealing with complex geometries by incorporating Monte Carlo (MC) methods. The method presented here is based on our publication [158]. This paper was published in Association for Computing Machinery (ACM) Transactions on Graphics (TOG) and presented at the ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH) Asia 2022 conference. For more information on my contribution to this work, please refer to Section 1.2.2.

Damien Rioux-Lavoie, Ryusuke Sugimoto, Tümay Özdemir, Naoharu H Shimada, Christopher Batty, Derek Nowrouzezahrai, and Toshiya Hachisuka. A Monte Carlo method for fluid simulation. *ACM Transactions on Graphics (TOG)*, 41(6):1–16, 2022

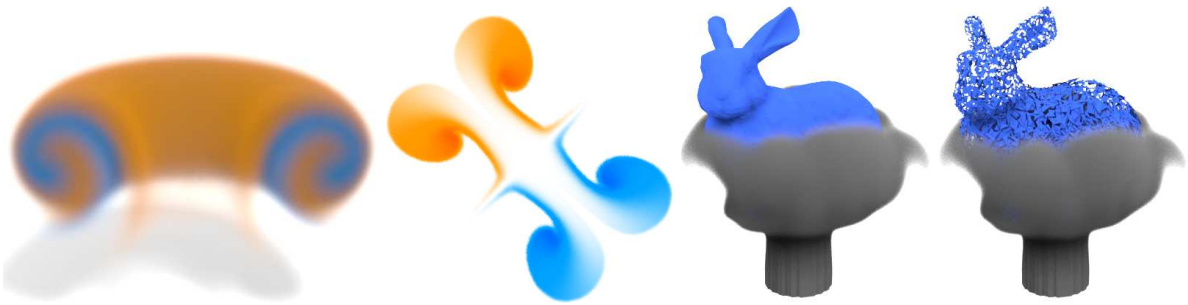


Figure 5.1: MCFluid. Our novel Monte Carlo fluid simulator supports two-dimensional and three-dimensional simulations of viscous incompressible flows by computing pointwise stochastic solutions of the vorticity transport equation. It is easy to implement and treats diverse fluid effects, such as leapfrogging vortex rings (left) and colliding jets (middle). The adoption of a Monte Carlo method to handle boundaries is well-suited to treating nontrivial boundary geometry, like a messy triangle soup (right).

5.1 Introduction

MC methods have been successfully applied to a diversity of problems in, e.g., statistical inference, simulation and integration [121].

In graphics, MC has been used most extensively in physically-based light transport where it outpaced radiosity-based traditional finite element methods to treat challenging radiometric effects with increasingly complex geometric and reflectance models [83, 153]. Recent applications of MC to problems in geometry processing [165] further evidence its broader applicability in graphics.

Motivated by this growing adoption, we adapt and demonstrate the utility of MC methods in computational fluid dynamics. We present a new Monte Carlo integration (MCI) approach to generate fluid motions with a pointwise stochastic formulation of solutions to the incompressible Navier–Stokes (NS) equations. We leverage a reinterpretation of the deterministic fluid equations as a stochastic process, introducing MC integral estimators of the Biot–Savart (BS) law that relates fluid vorticity and velocity fields. This reformulation allows us to define a recursive fluid flow model with striking similarities to simulation and sampling strategies employed in MC rendering.

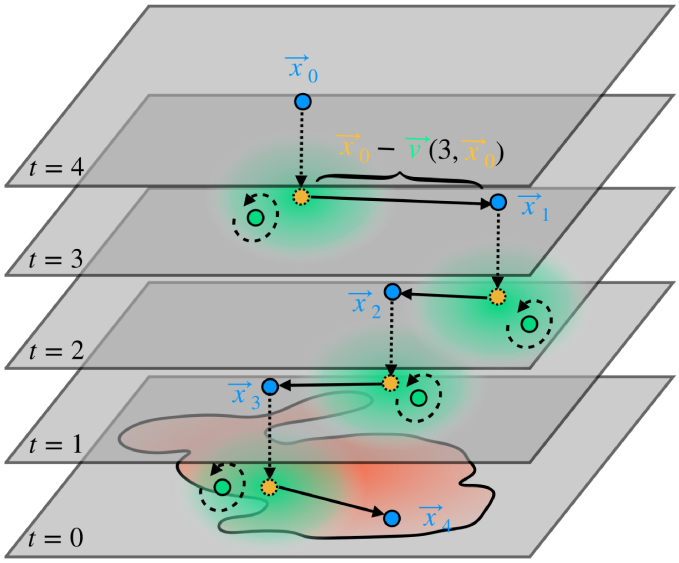


Figure 5.2: MC backtracing. To compute vorticity at a point and time of interest (in blue, top $t = 4$ layer) we project the point back to the previous time step (yellow) and compute its velocity with our BS estimator. To do so, we generate random samples (green) around the projected points, recursively computing their vorticity. Once computed, we advect along the velocity to the next position and repeat until we reach the initial condition.

For a throughout review of the relevent Monte Carlo methods, please refer to Section 3.1.3, Section 3.1.2.3 and Section 3.1.2.5. For the NS equations and the BS law, refer to Section 3.3.

Concretely, we first devise a recursive numerical integration scheme to solve two-dimensional (2D) incompressible Euler equations on open and periodic domains. Next, we develop a stream function-based strategy to enforce free-slip boundary conditions for static or moving obstacles, leading to a Poisson equation that we solve stochastically via Walk-on-Spheres (WoS) [131]. Our solver does not perform any global solve or boundary discretization, enabling a pointwise estimation and treatment of complex boundaries. Our stream function-based approach also allows for the treatment of inflow and outflow conditions on the domain boundary. Lastly, we generalize the method to three-dimensional (3D) NS equations (i.e., accounting for viscosity and

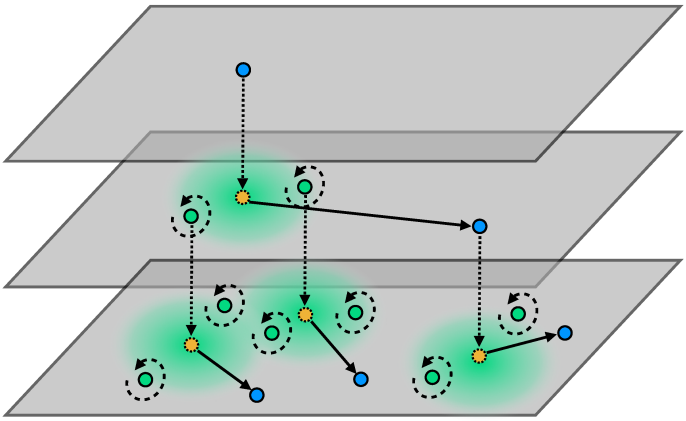


Figure 5.3: Exponential complexity. Applying our basic recursive formulation leads to exponential cost as, starting from a point of interest (blue, top), we must trace back for every random sample generated during the BS approximation (green), leading to an exponential branching factor.

stretching) using the Feynman–Kac (FK) formula that expresses partial differential equation (PDE) solutions as an expectation of a stochastic process, for which we compute with MC.

In its simplest form, our recursive formulation (Figure 5.2) exhibits exponential computation complexity (Figure 5.3)—akin to how distribution ray tracing scales for the number of light bounces. We overcome this problem with a practical uniform grid-based cache. Our caching scheme further allows us to develop and apply MC variance reduction techniques, including importance sampling (IS) of the vorticity field, and a control variates (CV) approach that utilizes the vorticity and velocity fields from the previous time step, to increase sample efficiency and accelerate stochastic estimates.

We validate our method against standard grid- and particle-based solvers, exploring and summarizing its behavior under different boundary conditions and parameter settings. The pointwise nature of our method allows an easy parallelization of computation. Our work is the first foray in the space of MC methods applied to fluid simulation in graphics and, in addition to highlighting the current limitations of our

proof-of-concept simulators, we outline and discuss a series of open problems associated to scaling MC-based fluid simulators to larger and more challenging flows.
Concisely, our contributions are:

- a novel MC fluid solver using recursive, pointwise probabilistic solutions to the 2D Euler equations based on the BS law,
- a WoS treatment of inflow, outflow, and free-slip solid boundary conditions using stream functions,
- a generalized MC solver for the full 3D incompressible NS equations based on the FK PDE formulation,
- a practical, non-recursive cache-based solver,
- applications of MC variance reduction to fluid simulation, and
- a roadmap of open challenges for scalable MC fluid simulation.

5.2 Basic Method

Fluid motion is commonly modeled by the incompressible Euler or NS equations describing the evolution of the ***velocity field*** $\vec{v}$ in time. By considering the ***vorticity field*** $\vec{\omega} \equiv \nabla \times \vec{v}$, one can instead derive a time-evolution equation for vorticity [34] that is the basis of various Lagrangian, Eulerian, and hybrid fluid solvers [41, 148, 172]. We adopt this vorticity-based approach, albeit departing substantially from the typical formulations.

Vorticity advection is determined uniquely by a flow's velocity and vorticity fields. In contrast to Lagrangian vorticity methods which propagate discrete vorticity elements forward in time, we will use a semi-Lagrangian-like interpretation that conceptually traces "backward in time" along the flow to determine the vorticity at a query point. To determine the required paths through the flow at any instant in time, one can reconstruct the velocity field from the vorticity field according to the BS law. We will apply MC to

this problem. The combination of backward tracing and MC yields our base recursive MC estimator of fluid flow (see Figure 5.2).

5.2.1 Vorticity Equation

Given a velocity field $\vec{v}$ and corresponding vorticity field $\vec{\omega}$, we start by recalling the well-established ***vorticity transport equation*** (Equation (3.134)) [34] for incompressible flow in three dimensions:

$$\frac{D\vec{\omega}}{Dt} = (\vec{\omega} \cdot \nabla)\, \vec{v} + \nu \nabla \cdot \nabla \vec{\omega}\,, \tag{5.1}$$

where ν is the ***kinematic viscosity*** coefficient and $D/Dt = \partial/\partial t + (\vec{v} \cdot \nabla)$ is the ***material derivative***. We assume constant density and viscosity, no external forces, and we defer discussion of boundary conditions and external forces to later. Temporarily assuming zero viscosity and considering the case of a 2D fluid, the so-called ***vortex stretching term*** $(\vec{\omega}\cdot\nabla)\, \vec{v}$ and the diffusion term are also zero, so it simplifies to the ***vorticity Euler equation***:

$$\frac{D\omega}{Dt} = 0\,, \tag{5.2}$$

which corresponds to pure advection of vorticity. Furthermore, the absence of a third dimension reduces the vorticity vector field to a scalar field $\omega = \frac{\partial}{\partial x} v_y - \frac{\partial}{\partial y} v_x$. This advection depends implicitly on the velocity field, which is in turn determined by the vorticity: the apparent simplicity of this evolution equation can thus deceptively misrepresent the fact that it models much more than a simple/passive "one-way" advection process.

In what follows, we denote the dependence of field quantities on time t and position $\boldsymbol{x}$ with parentheses, e.g., $\vec{\omega}(t, \boldsymbol{x})$, and we use the following notational conventions in 2D:

$\nabla \times \Psi = [\partial_y \Psi, -\partial_x \Psi]$, $\Psi \times \vec{A} = [-\Psi A_y, \Psi A_x] = -\vec{A} \times \Psi$ and $\nabla \times \vec{A} = \partial_x A_y - \partial_y A_x$, where Ψ is a scalar field and $\vec{A} = [A_x, A_y]$ is a 2D vector field.

5.2.2 Semi-Lagrangian Approach

Analytical solutions to Equation (5.2) are not available in the general case. One popular family of numerical methods to solve such advection problems is semi-Lagrangian schemes [178, 47]. After discretizing in time with time step Δt, a basic ***semi-Lagrangian scheme with Forward Euler time integration*** (Equation (3.139)) of trajectories approximates the updated vorticity $\vec{\omega}(t, \boldsymbol{x})$ at position $\boldsymbol{x}$ by querying the previous time step vorticity field $\vec{\omega}(t - \Delta t, \boldsymbol{x})$ as

$$\vec{\omega}(t, \boldsymbol{x}) \approx \vec{\omega}(t - \Delta t, \boldsymbol{x} - \vec{v}(t - \Delta t, \boldsymbol{x})\, \Delta t)\,. \tag{5.3}$$

5.2.3 Biot–Savart Law

We can advect the vorticity field $\vec{\omega}(t, \boldsymbol{x})$ based on Equation (5.3) given the velocity field $\vec{v}(t, \boldsymbol{x})$, which we in turn derive from $\vec{\omega}(t, \boldsymbol{x})$ using the ***BS law*** (Equation (3.136)) [34] that we recall here:

$$\vec{v}(t, \boldsymbol{x}) = \int_{\mathcal{X}} \vec{\omega}(t, \boldsymbol{y}) \times \vec{G}(\boldsymbol{x} - \boldsymbol{y})\, \mathrm{d}\boldsymbol{y}\,, \tag{5.4}$$

where $\mathcal{X}$ is a domain on which $\vec{\omega}$ is non-zero and the kernel is $\vec{G}(\boldsymbol{x} - \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})/(2\pi|\boldsymbol{x} - \boldsymbol{y}|^2)$ in 2D and $\vec{G}(\boldsymbol{x} - \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})/(4\pi|\boldsymbol{x} - \boldsymbol{y}|^3)$ in 3D. The velocity field obtained from Equation (5.4) satisfies both the divergence free property ($\nabla \cdot \vec{v} = 0$) and the definition of vorticity ($\vec{\omega} = \nabla \times \vec{v}$). Closed form solutions to Equation (5.4) are usually unavailable and we choose to estimate it with MCI.

5.2.4 Monte Carlo Integration

MCI (Section 3.1.3) is a stochastic numerical method to approximate integrals as a carefully weighted average of independent random samples. We make use of this method to approximate the BS integral (Equation (5.4)) to compute the velocity $\vec{v}(t, \boldsymbol{x})$.

Biot–Savart MC estimator

Let $\{\boldsymbol{y}_i\}_{i=1}^{n_{\text{mc}}}$ be n_{mc} independent samples drawn from probability distribution $p(\boldsymbol{y} \mid t, \boldsymbol{x})$. The MC estimator of Equation (5.4) is

$$\langle \vec{v}(t, \boldsymbol{x}) \rangle = \frac{1}{n_{\text{mc}}} \sum_{i=1}^{n_{\text{mc}}} \frac{\vec{\omega}(t, \boldsymbol{y}_i) \times \vec{G}(\boldsymbol{x} - \boldsymbol{y}_i)}{p(\boldsymbol{y}_i \mid t, \boldsymbol{x})} \,. \tag{5.5}$$

5.2.5 Monte Carlo Fluids

Building atop these aforementioned ideas, our central goal is to calculate the vorticity $\omega(\bar{t}, \bar{\boldsymbol{x}})$ of a dynamic fluid at simulation location $\bar{\boldsymbol{x}}$ and time $\bar{t}$. Note that we introduce this bar notation to emphasize that these are not variables, but fixed positions in space and time. The fluid has initial conditions $\omega(0, \boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^2$. Any subsequent state $\omega(\bar{t}, \bar{\boldsymbol{x}})$ is necessarily a function of the initial state $\omega(0, \boldsymbol{x})$ since the vorticity equation is deterministic. We begin by expressing this evolution function explicitly.

Recursive Integral Formulation: Combining the semi-Lagrangian scheme (Equation (5.3)) for time evolution and the BS law (Equation (5.4)), we represent $\omega(\bar{t}, \bar{\boldsymbol{x}})$ as an integral formulation of $\omega(\bar{t} - \Delta t, \bar{\boldsymbol{x}}')$ at a previous time step:

$$\omega(\bar{t}, \bar{\boldsymbol{x}}) \approx \omega(\bar{t} - \Delta t,\, \bar{\boldsymbol{x}} - \vec{v}(\bar{t} - \Delta t,\, \bar{\boldsymbol{x}})\, \Delta t) \tag{5.6}$$

$$= \omega\left(\bar{t} - \Delta t,\, \underbrace{\bar{\boldsymbol{x}} - \Delta t \int_{\mathcal{X}} \omega(\bar{t} - \Delta t,\, \boldsymbol{y}) \times \vec{G}(\bar{\boldsymbol{x}},\, \boldsymbol{y})\, \mathrm{d}\boldsymbol{y}}_{\bar{\boldsymbol{x}}'}\right). \tag{5.7}$$

We can represent $\omega(\bar{t} - \Delta t, \bar{\boldsymbol{x}}')$ as an integral equation of $\omega(\bar{t} - 2\Delta t, \bar{\boldsymbol{x}}'')$, backtracing recursively in time until reaching the initial condition $\omega(0, \boldsymbol{x})$. That is, $\omega(\bar{t}, \bar{\boldsymbol{x}})$ can be represented as a multiply-nested recursive integral formulation of $\omega(0, \boldsymbol{x})$. A similar multiply-nested recursive formulation is used in rendering to derive the *path space* integral formulation of light transport [153] in Section 3.2.4.4 and Section 3.2.4.7.

Monte Carlo Backtracing: We can now replace the analytical integral in Equation (5.7) with the MC estimator (Equation (5.5)) to solve the recursive integral by tracing backward in time to the initial conditions. While we discretized the time axis similarly to previous approaches in fluid simulation, this estimator provides a pointwise estimation of velocity without explicit discretization in space.

Consider an illustrative ***MC backtracing*** scenario in Figure 5.2. For simplicity we assume that $\Delta t = 1$. In order to compute the vorticity at a point of interest $\boldsymbol{x}_0$ (blue) and a given time $t = 4$ (top), we compute the velocity at this location in the previous time step (yellow) with our BS MC estimator. Doing so requires us to generate random samples (green) around the projected point and compute their vorticities recursively (Figure 5.3). Upon completing the recursion, we advect the point of interest to its backward position by the velocity and proceed recursively until we reach the initial condition of the system (at $t = 0$).

Discussion: To our knowledge, MC backtracing is the first numerical method to present a *pointwise* solution of the fluid equations both in space and time. Given the benefits of MC—namely, the ability to trade spatial discretization errors typical of alternative solvers for variance in the MC estimate and the capability to apply various variance reduction techniques—MC backtracing may be an attractive alternative. Moreover, when combined with physically-based MC rendering methods (e.g., path tracing), the variance in the fluid solver *and* the renderer may likely lead to more perceptually acceptable errors [33]. One important caveat in the formulation described thus far is that the recursive evaluations of

the vorticity at each backtraced sample lead to an *exponential* computational complexity (Figure 5.3). This major problem is implicitly woven into our formulation but we will devise solutions that alleviate it.

5.3 Advanced Methods

We present several extensions of our basic method to tackle a broader range of fluid problems and address computational constraints. We first introduce support for free-slip solid boundary conditions through an adaptation of the WoS algorithm using a stream function formulation. We then extend our formulation and method from the 2D Euler equations to the fully 3D incompressible NS, using the FK reformulation of the solution as a forward/backward stochastic process. Finally, we devise a more practical caching-based extension of the MC backtracing algorithm that overcomes the exponential cost of the base algorithm.

5.3.1 Boundary Conditions

When no boundary conditions (Section 3.3.1.1) are mentioned, we assume infinitely far or periodic boundaries and apply the BS velocity estimator (Equation (5.5)). To handle free-slip (or no through) solid boundary conditions, we employ a Poisson equation (with Dirichlet boundary conditions) to convert vorticity into a *stream function*, the curl of which yields the desired velocity.

Free-Slip Boundaries: We present our derivations in the 2D setting for clarity, but our ideas generalize fairly naturally to 3D (Section 5.3.2) using ***vector potentials*** in place of stream functions [19]. We initially consider boundaries with zero normal motion, i.e., $\vec{v}_{\text{wall}} \cdot \hat{n} = 0$. Under free-slip conditions, solid tangential velocity has no effect on the flow so we need not constrain it. We define the ***stream function*** $\Psi(t, \boldsymbol{x})$. We flip the usual sign

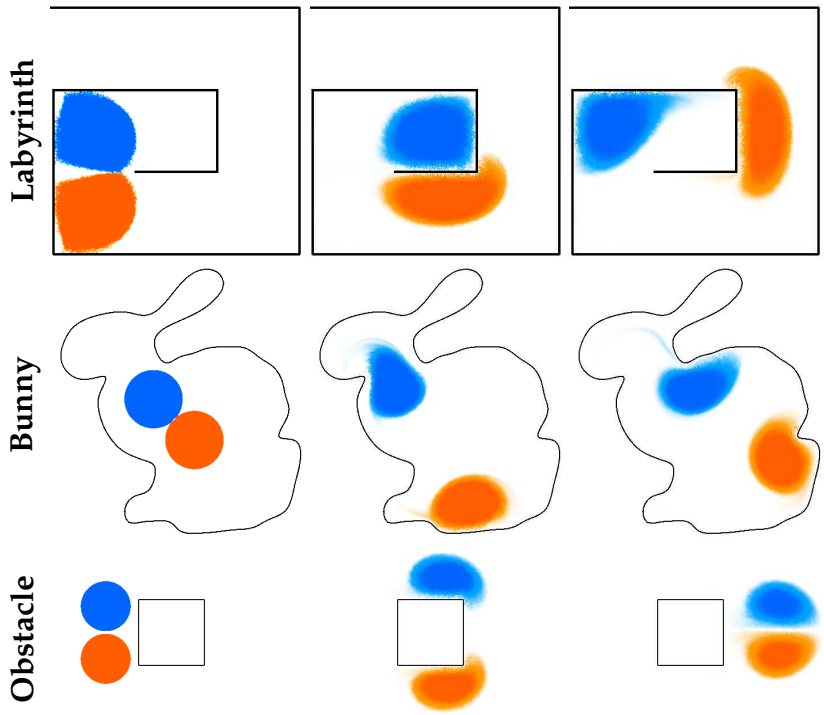


Figure 5.4: Free-slip boundary conditions. Left to right, $t = 0, 5, 10$ seconds. Two vortices propagating in a maze (top). A flow inside a more complex bunny-shaped boundary defined using a signed distance function (middle). Vortices flowing around a solid obstacle in a periodic domain (bottom).

convention [19] for the stream function ($\vec{v} = \nabla \times \Psi$) such that

$$\vec{v} = -\nabla \times \Psi \,. \tag{5.8}$$

Combining this expression with the definition of vorticity gives the following relationship (in 2D)

$$\omega = \nabla \times \vec{v} = \nabla \times (-\nabla \times \Psi) = \nabla \cdot \nabla \Psi \,. \tag{5.9}$$

From Equation (5.8), if Ψ is constant along the boundary (i.e., an isosurface of Ψ), the normal component of the velocity will be zero (i.e., $\vec{v} \cdot \hat{n} = 0$). Moreover, if the boundary is connected and piecewise smooth, we can arbitrarily set the isovalue to 0 without loss of generality (Figure 5.4). With multiple disconnected boundaries, we can still set the same

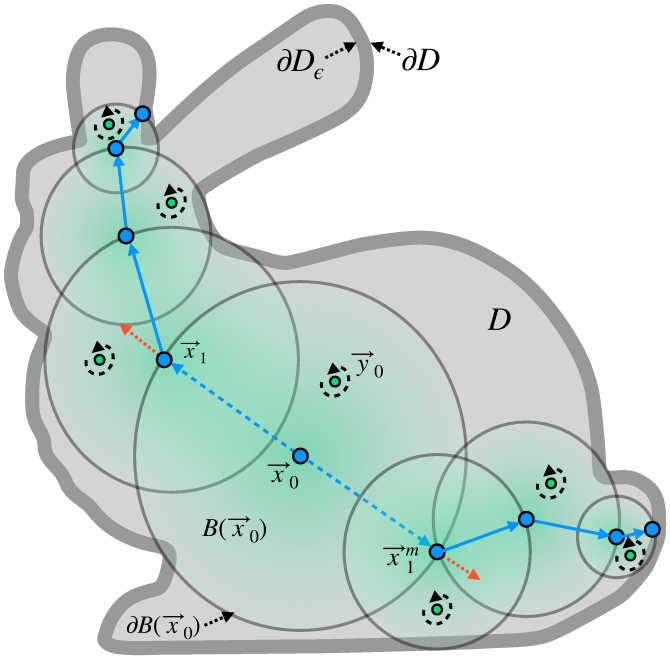


Figure 5.5: WoS gradient computation visualization. Consider the interior samples $\boldsymbol{y}_k \sim \partial B(\boldsymbol{x}_k)$ (green) and boundary samples $\boldsymbol{x}_{k+1} \sim B(\boldsymbol{x}_k)$ (blue). We sample two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_1^m$ on the largest inscribed sphere centered at $\boldsymbol{x}_0$ using antithetic sampling and compute their respective normals (red arrows) at the boundary. From these sampled positions, we proceed with a standard WoS—tracing two opposing random paths—and end the recursion once we penetrate a threshold region ∂D_ϵ (dark grey) to compute $\hat{\Psi}(\boldsymbol{x}_1)$ and $\hat{\Psi}(\boldsymbol{x}_1^m)$.

isovalue to each of them if we can expect zero net flow between the boundaries. Thus, after solving the Poisson equation (Equation (5.9)) for such a stream function we obtain a velocity field that satisfies our desired boundary conditions. In 3D, the true boundary condition $(\nabla \times \Psi) \cdot \hat{n}$ is more involved as it features derivative interactions among multiple components of the vector Ψ [1]. Like Bridson et al. [19], our current 3D results simply use $\Psi = \vec{0}$. Further study will be needed to properly resolve scenarios with multiple objects and nontrivial topologies.

Free-Slip Boundary Conditions

Given a domain $D \subset \mathbb{R}^2$, we impose slip boundary conditions using a stream function Ψ of the velocity field $\vec{v} = -\nabla \times \Psi$ by solving the following Poisson equation with Dirichlet boundary conditions

$$\nabla \cdot \nabla \Psi(\boldsymbol{x}) = \omega(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in D \text{ and} \tag{5.10}$$

$$\Psi(\boldsymbol{y}) = g(\boldsymbol{y}) \quad \text{for } \boldsymbol{y} \in \partial D\,, \tag{5.11}$$

where $g(\boldsymbol{y}) = 0$ and ∂D is the boundary of D.

Note that we do *not* first compute a current full-domain vorticity field and then solve this Poisson equation as a distinct step; rather, we use WoS to tightly integrate it into our recursive algorithm as a direct replacement for the BS MC velocity estimator (Equation (5.5)).

Walk-on-Spheres Method: The WoS method (Section 3.1.5) [131] is an MC pointwise solver for linear elliptic PDEs such as Poisson equations with Dirichlet boundaries. Since WoS computes the pointwise solution via MC sampling, it is well suited to our method.

Our interest is in determining the curl of the stream function (i.e., velocity), so we apply a gradient WoS estimator (Equation (3.78) and Equation (3.79)) [165] to the aforementioned Poisson equation with Dirichlet conditions, and use the components of $\nabla_{\boldsymbol{x}} \Psi$ to form the curl (e.g., in 2D by 90° rotation). To evaluate $\nabla_{\boldsymbol{x}} \Psi(\vec{x_0})$, the WoS algorithm estimates a recursive integral equation by recursively sampling a point $\boldsymbol{y}_k$ inside the largest sphere around the current point $\boldsymbol{x}_k$ and sampling another point $\boldsymbol{x}_{k+1}$ on the boundary of sphere to continue its recursion. A slight improvement is that we use antithetic sampling (Equation (3.63)) to get the first boundary sample, meaning that in addition to $\boldsymbol{x}_1$ we always add an extra sample, $\boldsymbol{x}_1^m$, on the opposite side of the sphere (also applying WoS to it); see Figure 5.5 for visual intuition. Omitting antithetic

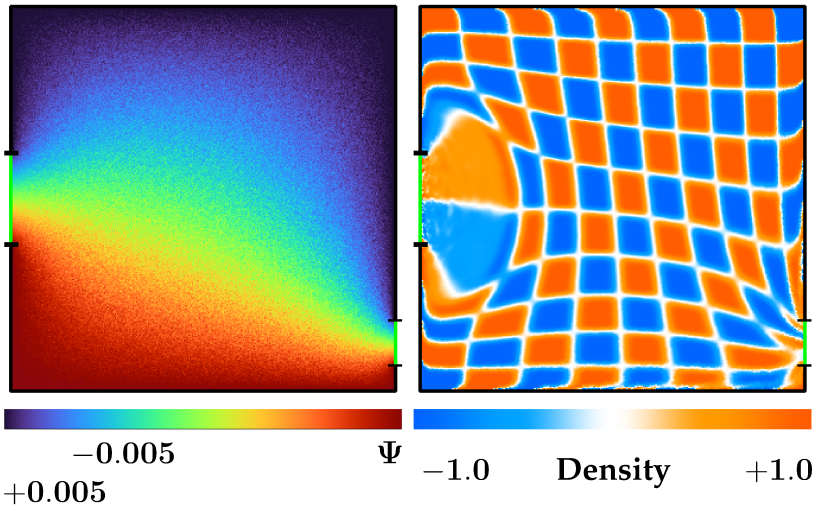


Figure 5.6: Inflow and outflow. Inflow and outflow regions highlighted in green on the boundary of the domain. Left: The stream function corresponding to setting the top part of the solid wall to $\Psi = 0.005$ and the bottom to $\Psi = -0.005$. Here, Ψ values at the inflow and outflow vary linearly, yielding constant normal velocity. Right: The motion of an advected checkerboard density field.

sampling can easily lead to a large variance when trying to compute the gradient of a constant non zero function.

Other Boundary Conditions: With a slight modification of the construction above, we can support ***inflow and outflow*** boundaries. By dividing the solid boundary into distinct pieces, each with a constant stream function value, and connecting them via linearly interpolated stream function values along the inflow/outflow segments of the boundary, we arrive at a stream function along the inflow/outflow whose gradient is parallel to the boundary (Figure 5.6). Taking the (negative) curl recovers the velocity that yields the desired perpendicular (constant) inflow or outflow.

We can further generalize our inflow/outflow treatment to moving ***solid boundaries with prescribed velocities***, again with free-slip (Figure 5.7). In 2D, the stream function

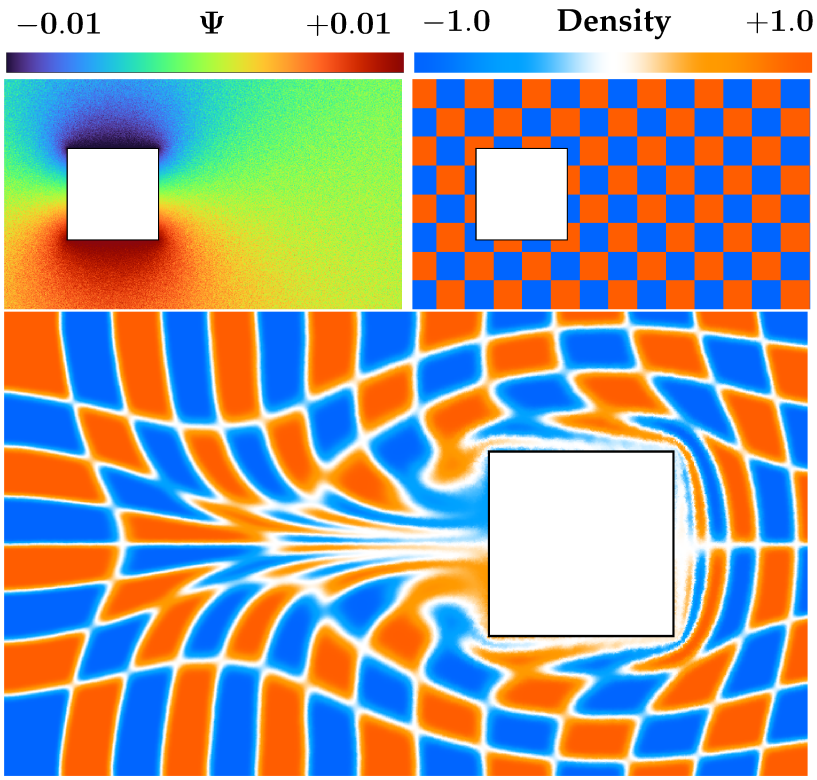


Figure 5.7: Moving obstacles. A square moves left to right at constant speed through the domain. The stream function (top left) and the initial density field (top right) to be advected, and the result after some time (bottom).

along the boundary must satisfy the constraint

$$\vec{v}_{\mathrm{b}} \cdot \hat{n} = \frac{\partial \Psi}{\partial \hat{t}}\,, \tag{5.12}$$

where $\vec{v}_{\mathrm{b}}$ is the velocity of the boundary, and $\hat{n}$ and $\hat{t}$ are the normal and tangent to the boundary.

We can choose any reference point on the solid surface and set it to some constant value Ψ_0, after which determining the stream function along the rest of the boundary (e.g., for a polygon with Ψ_i data at nodes $\vec{s}_i$) from the prescribed velocity field $\vec{v}_{\mathrm{b}}$ requires integrating the expression above to obtain

$$\Psi_{i+1} = \Psi_i + \int_{\vec{s}_i}^{\vec{s}_{i+1}} (\vec{v}_{\mathrm{b}} \cdot \hat{n})\, \mathrm{d}\vec{s}\,. \tag{5.13}$$

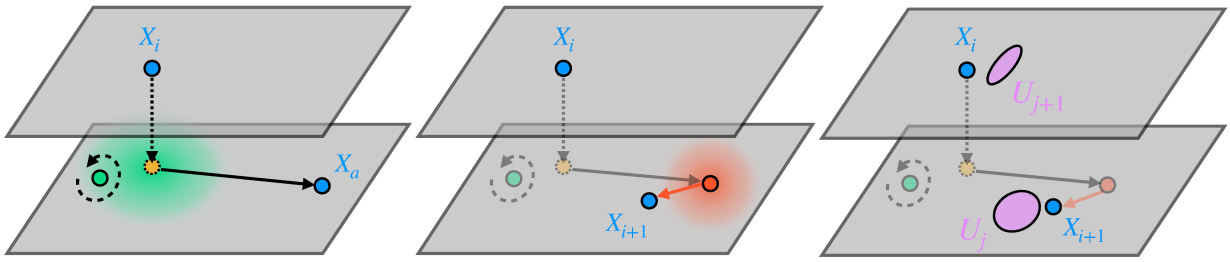


Figure 5.8: Recursive MCFluid solver. Left: First we advect the position of interest backward in time using a semi-Lagrangian approach and an MC velocity estimate. Middle: Then, we simulate viscosity by adding a Gaussian perturbation (with an appropriately-derived variance) to the advected position and compute the vorticity recursively. Right: Finally, we compute and propagate the stretching factor forward in time to the original position by multiplication with the computed vorticity.

Assuming a divergence-free solid boundary velocity field, the net flux along the surface will be zero; thus, the integration around its boundary loop back to $\boldsymbol{x}_0$ will yield the same starting value of Ψ.

For polygons and other simple shapes undergoing simple (e.g., rigid body) motions, we can obtain exact expressions of the stream functions on the obstacle boundaries [19].

5.3.2 Navier–Stokes Flows

Moving beyond the simple case of flows satisfying the 2D incompressible Euler equation, which we used to provide the intuition behind our backtracing algorithm, we now treat the more challenging case of incompressible NS flows in 3D. That is, we reintroduce the viscous diffusion and vortex stretching terms and rewrite the vorticity transport equation (Equation (5.1)) as

$$\frac{D\vec{\omega}}{Dt} = \mathcal{D}_{\vec{v}}\vec{\omega} + \nu\nabla\cdot\nabla\vec{\omega}\,, \tag{5.14}$$

where $\mathcal{D}_{\vec{v}} = (\nabla\vec{v} + \nabla\vec{v}^{\intercal})/2$ is the ***strain-rate tensor***. The new term $\mathcal{D}_{\vec{v}}\vec{\omega}$ is an alternative but equivalent form of the stretching term $(\vec{\omega}\cdot\nabla)\vec{v}$ that will be easier to work with.

Feynman–Kac Representation: We wish to compute solutions to Equation (5.14) at a specific time $\bar{t}$ and position $\bar{\boldsymbol{x}}$, given an initial condition on the vorticity field, analogous to the earlier 2D case. The problem now, however, is more complex: the viscous term effectively adds randomness to the possible trajectories of fluid particles and the stretching term will deform the vorticity according to the flow. To overcome these complexities, we take inspiration from theoretical work on the open problem of the existence and smoothness of solutions to NS [22] and the path integral formulation of quantum mechanics. We apply the FK formula (Section Section 3.1.2.5)—a mathematical tool that expresses the solution to deterministic PDEs as a stochastic process—to the fluid equations. Doing so will enable us to express our numerical solution as a weighted expectation of the initial vorticity, deformed along various trajectories. More formally, the ***FK formula*** ((3.51)) [142] states that solutions of the vorticity transport equation satisfy

$$\vec{\omega}(\bar{t}, \bar{\boldsymbol{x}}) = \mathbb{E}\left[U_{\bar{t}}\, \vec{\omega}(0, X_{\bar{t}}) \,|\, U_0 = I, X_0 = \bar{\boldsymbol{x}}\right] , \tag{5.15}$$

where the expectation is with respect to the Wiener measure, i.e., all possible realizations of the ***Wiener processes*** W_s (Section 3.1.2.2). Here, the ***Lagrangian path*** X_s is a realization of the time-reversed stochastic process Y_s defined by the stochastic differential equation (Equation (3.50))

$$\mathrm{d}Y_s = -\vec{v}(\bar{t} - s, Y_s)\,\mathrm{d}s + \sqrt{2\nu}\,\mathrm{d}W_s \text{ with } Y_0 = \bar{\boldsymbol{x}} , \tag{5.16}$$

and the ***deformation matrix*** along a reversed Lagrangian path X_s is

$$U_t = \exp\left(\int_0^t \mathcal{D}_{\vec{v}}\left(\tau, X_{t-\tau}\right)\mathrm{d}\tau\right) . \tag{5.17}$$

It has been shown that, given a Lagrangian path, this process (Equation (5.17)) is a solution to the differential equation

$$\mathrm{d}U_t = \mathcal{D}_{\vec{v}}\left(t, X_{\bar{t}-t}\right) U_t \,\mathrm{d}t \text{ with } U_0 = I\,. \tag{5.18}$$

We refer to Equation (5.16) as the ***backward process***, since it propagates backwards in time (i.e., $s = \bar{t} - t$), and to Equation (5.18) as the ***forward process***. These expressions and their derivations appear in [22] but have not been used to construct a practical numerical method.

While daunting at first glance, the intuition behind these expressions is comparatively straightforward, as illustrated in Figure 5.8. First, consider the backward process (Equation (5.16)) associated to a stochastic differential equation that models the backward trajectory of fictitious particles that cross the position $\bar{\boldsymbol{x}}$ at time $\bar{t}$. The process traces back the trajectories of these particles according to the underlying velocity field (left term, RHS) as we did before, but now additionally accounts for the diffusive process (rightmost term) through the randomness of the Wiener process W_s. All these trajectories carry some vorticity according to their initial state, however, these vorticities need also be warped by the local strain-rate tensor along their trajectory to account for vortex stretching. This is precisely what the forward process (Equation (5.18)) models: it stretches the initial particle vorticities according to the strain-rate along their trajectory. Finally, the FK formula (Equation (5.15)) states that the vorticity at location $\bar{\boldsymbol{x}}$ and time $\bar{t}$ *is exactly* the average over the stretched vorticity carried by all trajectories that crossed the position at the given time[3].

Monte Carlo Estimation: We construct a three-step algorithm to evaluate the expectation of Equation (5.15): an advection, a diffusion, and a stretching step. We adopt

[3]An alternative interpretation draws on the Feynman path integral in quantum mechanics, with probability weights assigned to paths and where the quantity of interest at a location is determined by the average over all possible probability-weighted paths.

the notation $X_i = X_{s_i}$ where $s_i = i\Delta s$, $\Delta s = \bar{t}/n$ and $i \in \{0, \cdots, n\}$ for the discretized quantities. We discretize Equation (5.16) with a semi-implicit variant of the ***Euler–Maruyama*** [116] method (Equation (3.48)), yielding the Lagrangian path

$$X_{i+1} = X_i - \overbrace{\Delta s\, \vec{v}(\bar{t} - s_{i+1}, X_i)}^{\text{Advection}} + \overbrace{\sqrt{2\nu\Delta s}\, \bar{\boldsymbol{\xi}}_i}^{\text{Diffusion}} \text{ with } X_0 = \bar{\boldsymbol{x}}\,, \tag{5.19}$$

where $\bar{\boldsymbol{\xi}}_i \sim \mathcal{N}(\vec{0}, I)$ is a normally distributed random sample. Without viscosity this discretization simplifies to that of Section 5.2. If one uses just a single MC sample during the diffusion step, our treatment of diffusion becomes essentially consistent with that of Chorin [26], where the position is perturbed using appropriately-scaled Gaussian noise.

Our derivation is actually agnostic to the choice of temporal discretization. For instance, the deterministic term of Equation (5.16) could instead be discretized using other standard advection approaches, such as Runge–Kutta 4th order (RK4) and MacCormack methods. However, for simplicity of presentation we will use the simple semi-implicit form given in the equations above.

We similarly discretize the deterministic Equation (5.18) with a forward Euler scheme, yielding

$$U_{j+1} = [I + \overbrace{\Delta t\, \mathcal{D}_{\vec{v}}\,(t_j, X_{\bar{t}-t_j})}^{\text{Stretching}}]\, U_j \text{ with } U_0 = I\,. \tag{5.20}$$

One can approximate the strain-rate tensor $\mathcal{D}_{\vec{v}}$ in various ways: by finite differences, by applying MCI by passing the gradient of the velocity field through the BS integral (Equation (5.4)) or through a Hessian WoS estimator, or by converting the vorticity field to a vortex segment representation and advecting it according to the velocity field, as in the work of Zhang and Bridson [207]. We adopt the last approach due to its superior

stability in practice; the term $\mathcal{D}_{\vec{v}}\vec{\omega}$ can be approximated by

$$\mathcal{D}_{\vec{v}}\vec{\omega}(t,\boldsymbol{x}) \approx \frac{|\vec{\omega}(t,\boldsymbol{x})|}{h}\left[\vec{v}\left(t,\boldsymbol{x}+\overrightarrow{\Delta x}\right)-\vec{v}\left(t,\boldsymbol{x}-\overrightarrow{\Delta x}\right)\right], \tag{5.21}$$

where $\overrightarrow{\Delta x} = (h/2)\,\vec{\omega}(t,\boldsymbol{x})$ and h is the length of the vortex segment. (Note that $\overrightarrow{\Delta x}$ is unrelated to the gradient of $\boldsymbol{x}$.) One can estimate the velocities at the ends of a vortex segment using the appropriate velocity estimator to get an estimate of $\mathcal{D}_{\vec{v}}\vec{\omega}$. See Section 5.3.3 for implementation details.

MCFluid

When computing the vorticity $\vec{\omega}$ at time $t_i = i\Delta t$ and position $\boldsymbol{x}_i$ with initial condition $\vec{\omega}(0,\cdot)$ known, we have

$$\vec{\omega}(t_i,\boldsymbol{x}_i) = \frac{1}{n_d}\sum_{j=1}^{n_d}\left[\vec{\omega}\left(t_{i-1},\boldsymbol{x}_{i-1}^j\right)+\Delta t\left\langle\mathcal{D}_{\vec{v}}\vec{\omega}(t_{i-1},\boldsymbol{x}_{i-1}^j)\right\rangle\right],$$
$$\boldsymbol{x}_{i-1}^j \sim \mathcal{N}(\boldsymbol{x}_i-\Delta t\left\langle\vec{v}(t_{i-1},\boldsymbol{x}_i)\right\rangle, \sqrt{2\nu\Delta t}\,I), \tag{5.22}$$

where n_d is the number of diffusion samples, $\langle\vec{v}\rangle$ is the MC estimate of the velocity using either the BS or WoS estimators (Equations (5.5), (5.8)), and $\langle\mathcal{D}_{\vec{v}}\vec{\omega}\rangle$ is the estimate of the stress rate, i.e., stretching term (Equation (5.21)). We use the same coloring scheme to represent the different steps (advection, diffusion, and stretching) of Algorithm C.1.

Implementation: After rewriting the FK expectation (Equation (5.15)) as a one-step MC estimate and replacing the velocity and the stretching term with their respective MC estimates, we arrive at our algorithm (see pseudocode in Algorithm C.1).

5.3.3 Practical Implementation with Caching

As outlined earlier, a naive implementation of our MCFluid algorithm (Equation (5.7)) suffers from an exponential increase in vorticity samples—all of which must be recursively evaluated—with respect to time (Figure 5.3). We offer a strategy to circumvent this problem using a uniform grid cache. Similar to dynamic programming solutions to recursive problems, we store and reuse the vorticity field (and the velocity field in some applications, as we discuss later) as it is computed. We can proceed in two ways.

The first is to fill a spatio-temporal grid using the same recursion as before with respect to a position and time of interest. Doing so will gradually fill out a spatio-temporal "cone" of earlier data, rather than evaluating over the whole domain for all time steps. This approach may be ideal when one wishes to evaluate the simulation backwards in time, e.g., in view-dependent simulation.

Alternatively, we can use only a single spatial grid that stores the vorticity computed at the center of each cell at the preceding time step. In essence, at every time step we query this cache to get the previous time step vorticity during MC estimation of the BS integral for the velocity field. As we will discuss later, this method also conveniently offers a way to generate importance sampled MC samples distributed according to the vorticity field, reducing variance and increasing sample efficiency. Unless noted otherwise, all results are generated using this latter approach.

To evaluate the stretching term (Equation (5.21)) for 3D simulations, we choose h to be the grid resolution in the absence of boundaries, and in the presence of boundaries we use the minimum of the grid resolution and twice the closest distance to any boundary. For this stretching term computation, we fetch the cached velocities in practice because the bilinear or trilinear interpolation smoothes out the velocity field, which reduces the noise when we compute the stretching term, and because we can save computational time.

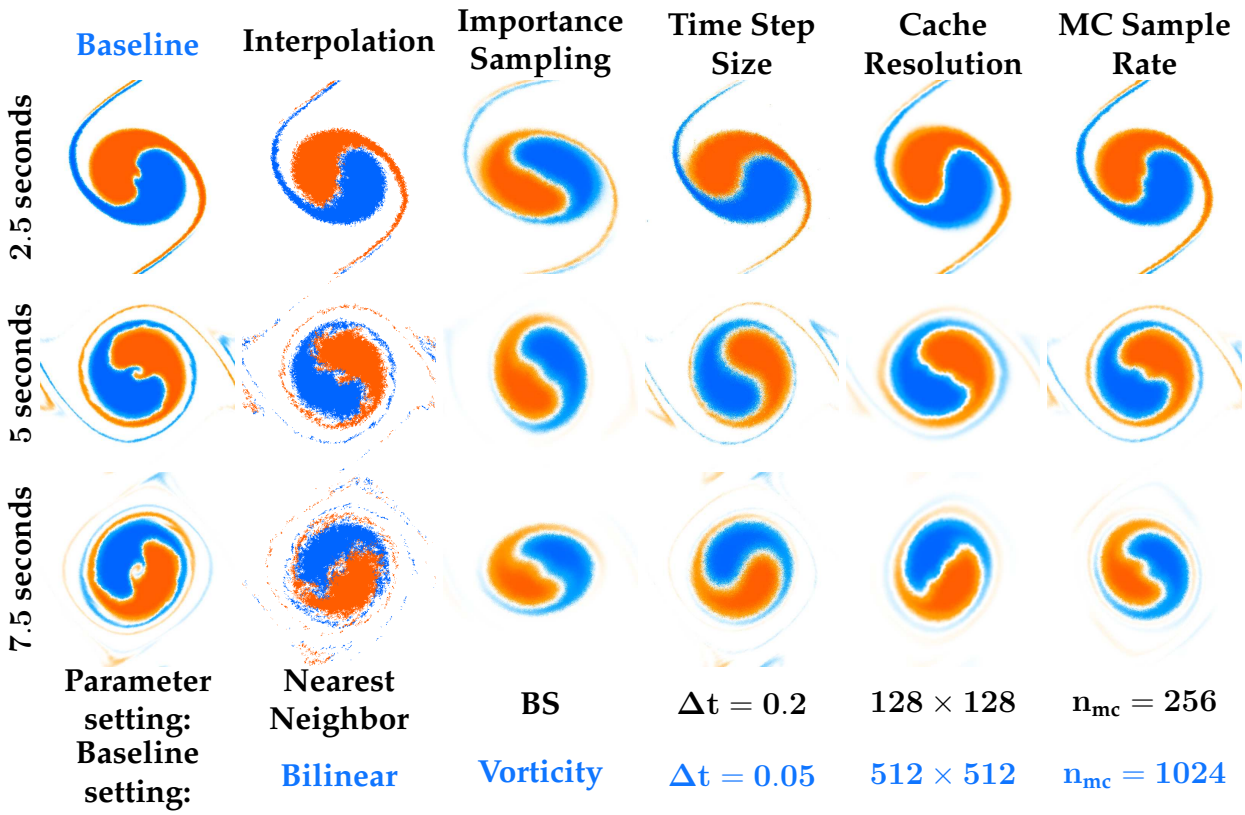


Figure 5.9: Parameter settings. Top to bottom, frame times are $t = 2.5, 5, 7.5$ seconds. Our baseline simulation (left) with high quality parameter settings (in blue) uses a 512×512 uniform cache resolution, IS of the vorticity field, bilinear interpolation, $\Delta t = 0.05$ seconds, $n_{\text{mc}} = 1024$ samples, and $\nu = 0$. Subsequent columns analyze the impact of adjusting exactly one parameter in favor of a faster but less accurate result. Left to right: baseline/high-quality setting, nearest neighbor interpolation, IS the BS kernel, larger time step, coarser cache, and lower MC sampling rate.

5.3.4 Variance Reduction

Our MC approach opens the opportunity to exploit a plethora of *variance reduction methods* within the context of fluid simulation. Indeed, we have already presented the antithetic sampling method for the WoS estimator to reduce the variance. We additionally present a few more basic yet practical variance reduction methods as examples of what would be possible with our MC approach.

Importance Sampling: One of the most popular and basic variance reduction methods is IS (Section 3.1.3.2). The idea is to generate samples according to a given probability density function (PDF) to reduce the variance. A well chosen PDF can drastically reduce the variance, and thus the error of the estimators, and we can come up with a couple of such PDFs that are well suited to our setting.

The first option is to generate samples proportionally to the integral kernel; the BS kernel in the BS case (i.e., infinite/periodic boundaries) and the harmonic Green's function in the WoS case (i.e., prescribed boundaries). Sampling according to either of those functions is trivial. This IS works especially well for WoS, but not always for the BS case.

For the BS case, this PDF can be only marginally better than uniform sampling in case non zero vorticities are concentrated in a small region. The second option in this case is to importance sample directly according to the magnitude of the vorticity field. Since we are already caching vorticities to overcome the exponential cost of recursion, we can utilize this cache to perform IS. Figure 5.9 illustrates an equal sample comparison between these two options. In this example, the baseline which uses sampling according to vortices produced a more accurate result (e.g., long and thin features of vorticities) than the alternative of sampling according to the BS kernel.

Control Variates: A sequential nature of fluid simulation fits well to the method of CV (Section 3.1.3.2). The method of CV utilizes another analytically integrable function (a CV) to estimate only the difference between this function and the original integrand via MCI. This method can reduce variance when the original integrand and the CV are correlated.

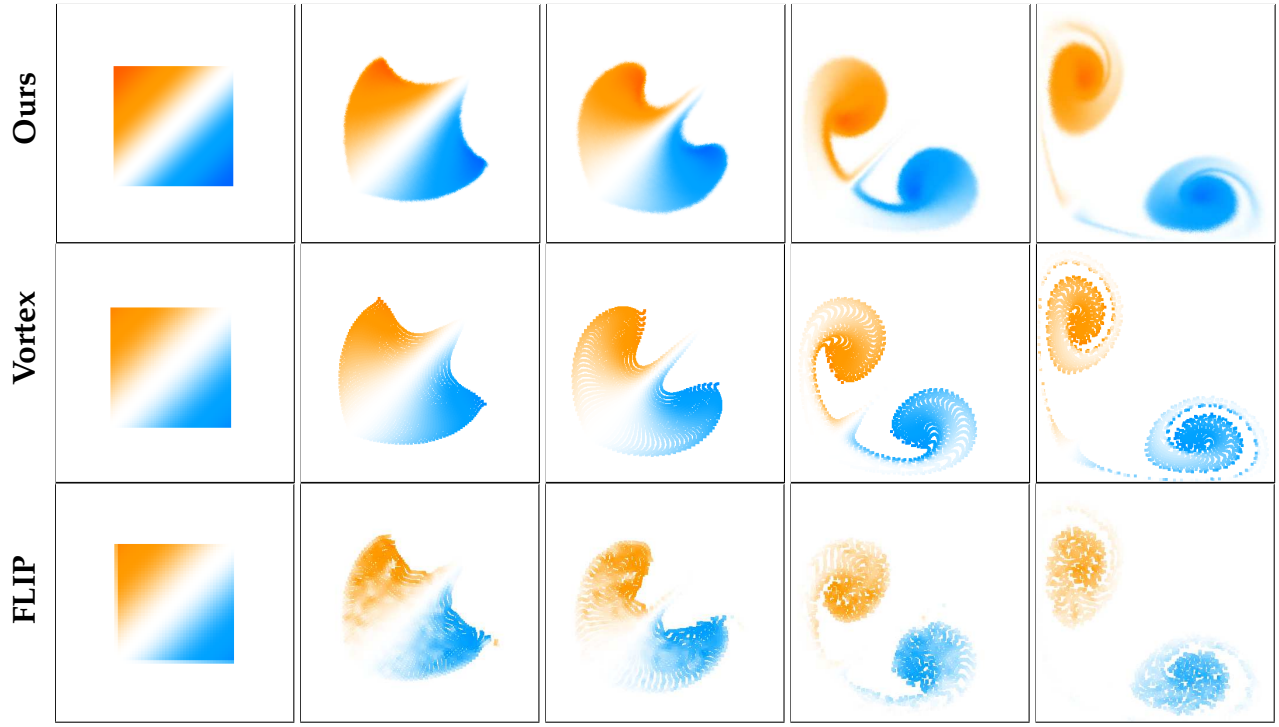
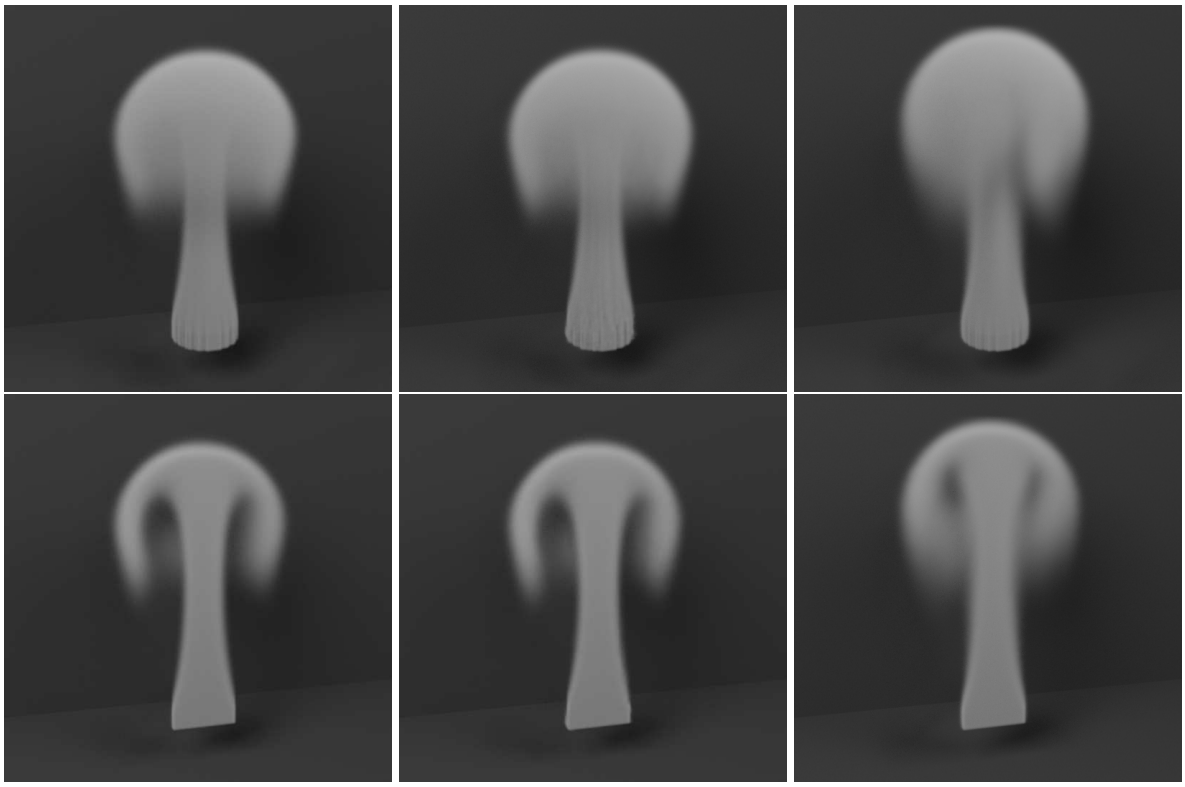


Figure 5.10: Comparison with standard solvers. Left to right, $t = 0, 1, 2, 5, 10$ seconds. Our method (top) produces results that are consistent with the vortex particle (middle) and hybrid fluid-implicit-particle (FLIP) (bottom) baselines. Colors visualize the signed vorticity at each (particle) position.

In our application, to estimate the velocity using the BS law (Equation (5.4)), we can use *the vorticity field from the previous time step* as a CV, as follows:

$$\begin{aligned}\vec{v}(t,\boldsymbol{x}) &= \int_{\mathcal{X}} [\vec{\omega}(t,\boldsymbol{y}) - \vec{\omega}(t-\Delta t,\boldsymbol{y})] \times \vec{G}(\boldsymbol{x}-\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} \\ &\qquad + \int_{\mathcal{X}} \vec{\omega}(t-\Delta t,\boldsymbol{y}) \times \vec{G}(\boldsymbol{x}-\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} \\ &= \int_{\mathcal{X}} [\vec{\omega}(t,\boldsymbol{y}) - \vec{\omega}(t-\Delta t,\boldsymbol{y})] \times \vec{G}(\boldsymbol{x}-\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} + \vec{v}(t-\Delta t,\boldsymbol{x})\,.\end{aligned} \tag{5.23}$$

Note that having already estimated $\vec{v}(t-\Delta t,\boldsymbol{x})$ from $\vec{\omega}(t-\Delta t,\boldsymbol{y})$ in the previous time step, this term is available in the current time step. Unlike a typical application of the method of CV, however, this term $\vec{v}(t-\Delta t,\boldsymbol{x})$ is only an estimation with some variance, not an analytical integration of $\vec{\omega}(t-\Delta t,\boldsymbol{y})$. When $\vec{v}(t-\Delta t,\boldsymbol{x})$ is similarly estimated by using



(a) reference **(b)** with CV **(c)** w/o CV

Figure 5.11: Control variate. When the number of samples used is small, the application of CV (middle) produces the results closer to reference simulation results (left) compared to the results without CV (right). We advect a constant density at the bottom according to the velocity field simulated with our method, and render the results fully (top) and with a cross section visualization (bottom).

$\vec{\omega}(t - 2\Delta t, \boldsymbol{x})$ as the CV, this approach may not reduce variance overall. We circumvent this issue by estimating the initial velocity $\vec{v}(0, \boldsymbol{x})$ using a higher sample count than the rest. This approach will propagate variance reduction via this CV at $t = 0$ all the way to the current time, without increasing the sample count in any other time than $t = 0$.

We generate samples according to the difference of vorticity fields at two consecutive time steps to evaluate the first integral, and add the cached velocity from the previous time step. Due to the high correlation between the vorticity fields over time, we can expect that the variance of $\vec{\omega}(t, \boldsymbol{y}) - \vec{\omega}(t - \Delta t, \boldsymbol{y})$ is smaller than $\vec{\omega}(t, \boldsymbol{y})$. Thus, this method

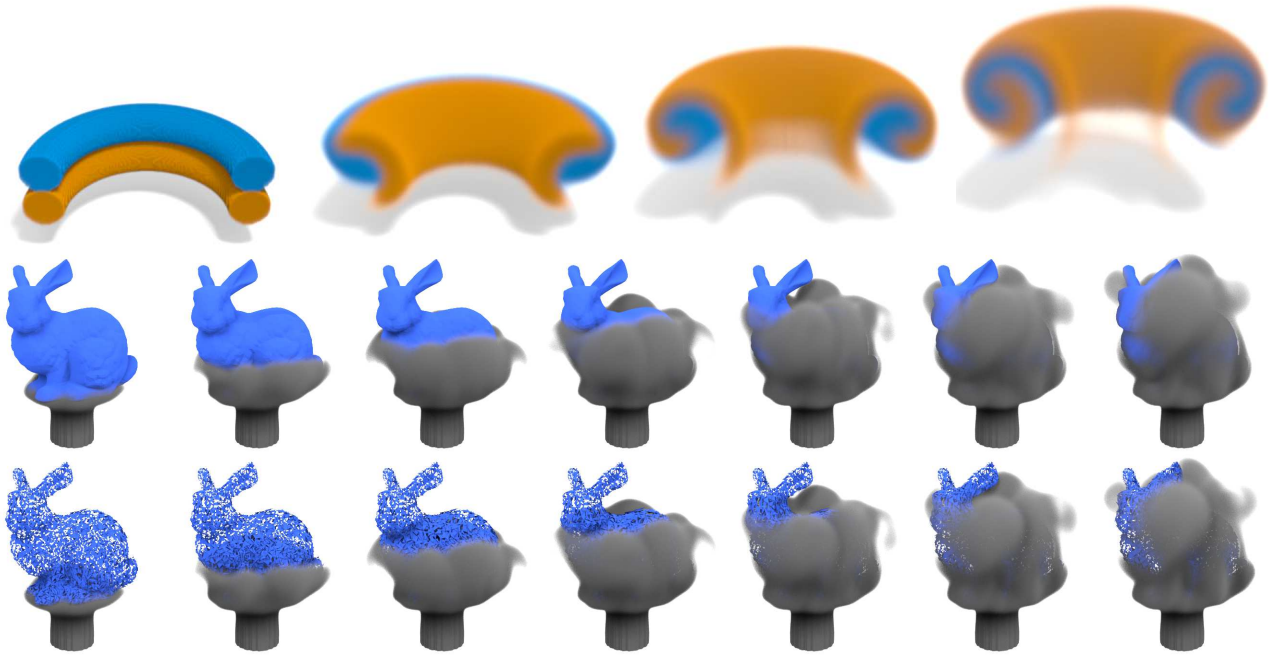


Figure 5.12: 3D simulations. We simulate two leapfrogging vortex rings (top) with a cross section visualization to illustrate the interior flow. We passively advect constant density fields toward closed mesh (middle) and triangle soup Stanford bunnies (bottom). We render all simulations with Blender's [2022] principled volume shader.

greatly reduces the variance of our velocity estimate, given that the variance of the initial velocity field is low enough. Figure 5.11 illustrates an equal sample comparison when we have the CV enabled and disabled, and the method of CV works well as expected. We applied this method only to the 3D scenes (without boundaries), where we expect significant reductions of sampling cost.

5.4 Results

Comparison to Existing Methods: We compare our method to a pair of representative existing methods (Figure 5.10): a particle-based solver that implements the vortex particle method of Park and Kim [148]—using the BS law to forward integrate vortex particle trajectories (2500 particles) while treating free-slip boundaries with a panel method (80 panels)—and a (hybrid) grid-based fluid-implicit-particle (FLIP) gas solver [209, 18] with

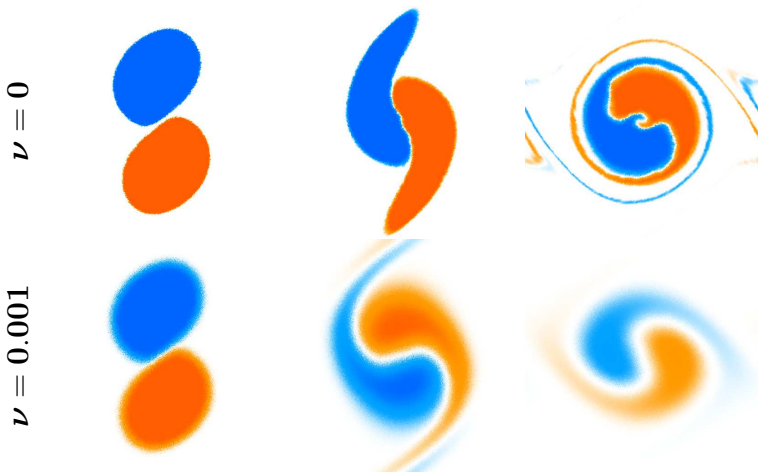


Figure 5.13: Viscous simulation. Left to right, $t = 0.2, 2, 5$ seconds. Inviscid simulation with $\nu = 0$ (top) and viscous with $\nu = 0.001$; $n_{\mathrm{d}} = 4$. (bottom)

grid resolution $n_{\mathrm{x}} = 50$, 6 FLIP particles per grid cell, and $\Delta t = 0.05$. We visualize the vorticity field, in blue (positive) and orange (negative). We initialize the methods with the same vector field. Our simulation output is in strong agreement with the two baselines, suggesting that it indeed generates a valid solution to the fluid equations.

Our method using WoS runs at an average of 21 seconds per frame (with a multi-threaded CPU implementation). Both the vortex particle method and FLIP run orders of magnitude faster (i.e., roughly 0.005 and 0.25 seconds respectively). Sawhney and Crane [165] also noted a similar performance gap in their comparisons between WoS and finite-element methods for geometry processing. Despite this significant room of improvement in computation time, our method does have some fundamental advantages. First, the lack of reliance on a linear solve and the pointwise nature of our method makes it easily parallelizable and graphics processing units (GPU) friendly (as evidenced by our ShaderToy and compute unified device architecture (CUDA) implementations, discussed below). Second, particle methods rely on nearest neighbor search to interpolate values which can become quite expensive with many particles, which does not exist in our method. Third, grid-based solvers are generally suffering from numerical diffusion, as evidenced in the fluctuations of colors in Figure 5.10, which

is less evident in our method due to its pointwise nature. Last, our formulation based on WoS enables handling of noisy boundaries with no additional effort, as we discuss later. Similar to its application in geometry processing [165], a further study and better implementation of our MC approach would reduce the performance gap, leaving the fundamental advantages of MC as noted above.

Viscosity: The viscosity formulation of our 3D flow solver (Section 5.3.2) applies equally well in 2D. Figure 5.13 presents a 2D viscous simulation using our approach and a 512×512 uniform cache, nearest neighbor interpolation, IS of the vorticity cache, $\Delta t = 0.05$, and $n_{\text{mc}} = 1024$. Given that physical diffusion has an effect similar to interpolation, we found nearest neighbor (rather than bilinear) interpolation to be acceptable here.

Boundary Conditions: Figure 5.4 presents various solid boundary configurations. All three scenes use a uniform cache of 512×512, $\Delta t = 0.05$, $n_{\text{mc}} = 256$, and $\nu = 0$. As our method for free-slip boundary conditions relies on WoS, it benefits from some of the same attractive properties as the method of Sawhney and Crane [165], e.g., flexibility in the choice of geometric representations, such as meshes, polygon soup, or even *unsigned* distance functions.

In Figure 5.6, we illustrate our inflow/outflow boundary conditions. Since the integral of the stream function on the boundary is zero, the corresponding velocities yield zero net flux; that is, exactly as much matter enters the domain as exits.

Figure 5.7 illustrates our moving boundary support. Here, we treat a constant translational velocity for simplicity, since this imposes a constant stream function on the boundary and simplifies computation. Obtaining analytic formula for complex rigid bodies undergoing simple motion is a possibility for more advanced applications, as noted in Section 5.3.1. Both the inflow and moving boundary application were implemented as high-performance real-time demos in the online GPU ShaderToy framework, evidencing the suitability of our approach to parallel computation.

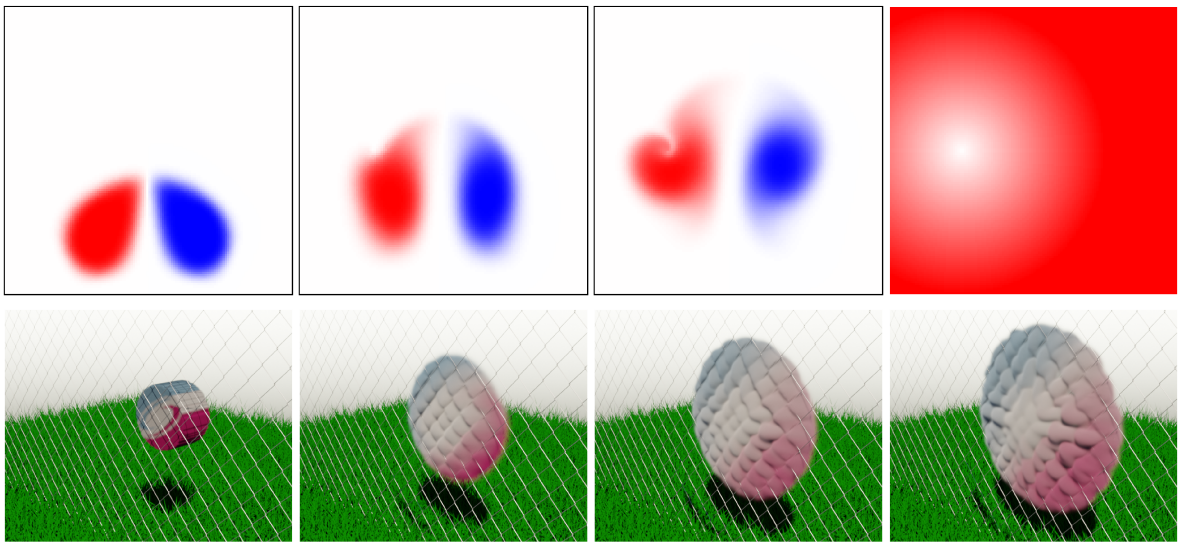


Figure 5.14: Subgrid-size obstacle. Our approach can take into account a tiny obstacle. A subgrid-sized square (the rightmost image visualizes its unsigned distance function) affects the fluid flow (top). In 3D, subgrid-width chain link fence wires affect the fluid flow (bottom).

Figure 5.12 (bottom) and Figure 5.14 show that our method can accept complex geometry without the added effort of other methods [7, 75, 111], such as challenging tetrahedral meshing or cut-cell generation for velocity-based schemes or panel methods in vorticity schemes. As emphasized by Sawhney and Crane [165], accurate conforming mesh generation for a complex obstacle can take many minutes or hours; however, as long as we have access to a method for fast evaluation of the distance field, our MC-based methods can immediately be applied. In the triangle soup boundary example in Figure 5.12, we construct a bounding volume hierarchy (BVH) tree of triangles to evaluate distance to the closest boundary. Figure 5.14 shows a case when the domain contains an obstacle whose size is smaller than the cache grid resolution (about 1/39 of a grid cell width for the 2D scene, and 1/3 of a cell width for the 3D scene), yet is naturally respected by the flow. A typical grid-based solver would either miss this geometry altogether, rely on conservative rasterization to non physically inflate the obstacle, or need to introduce an approximate drag model to influence the flow. Notably, the

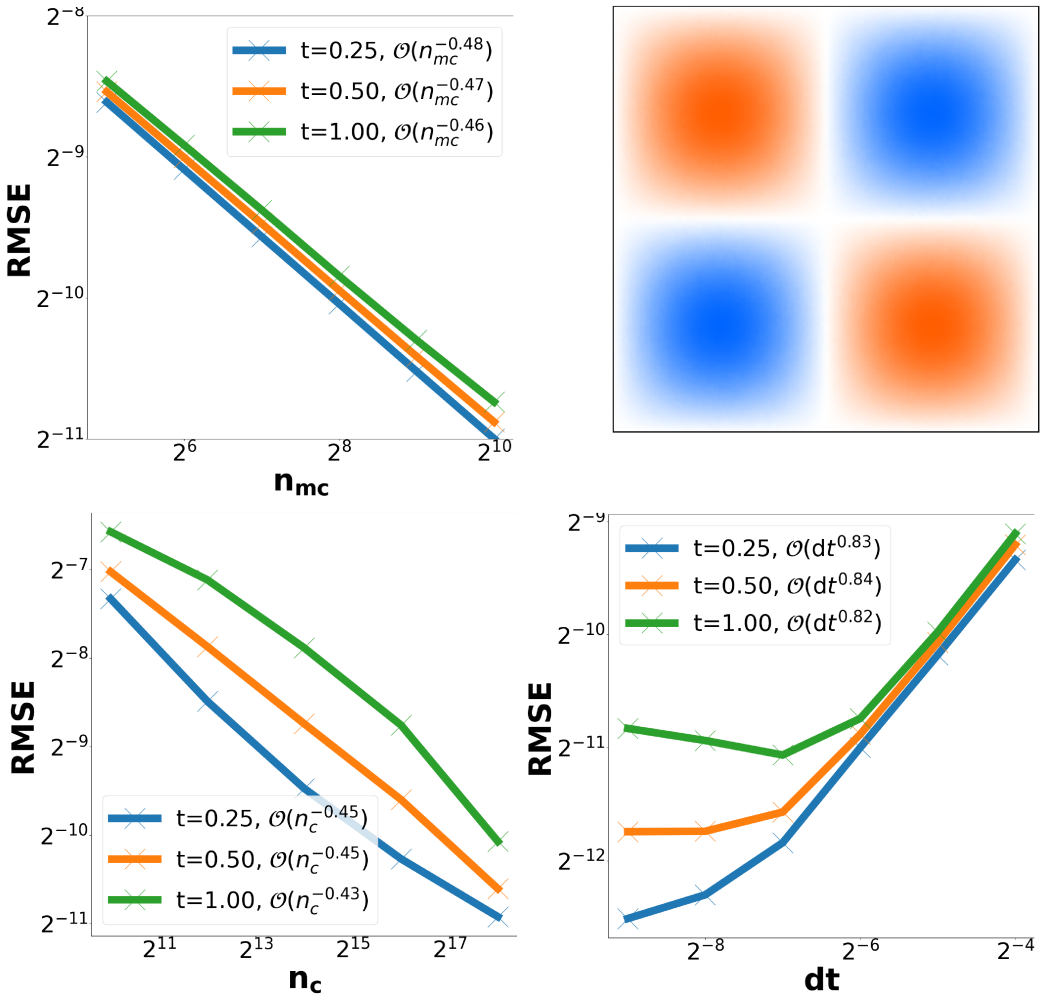


Figure 5.15: Convergence analysis. Root mean squared error computed against the analytical solution $\omega(x, y) = \sin(x)\cos(y)$ for $x, y \in [-\pi, \pi]^2$ (top right). We show the convergence profile of different parameters at different physical times $t = 0.25, 0.5, 1$ seconds. We observe that our method does converge numerically to the true solution as the number of cells $n_c = n_x^2 \to \infty$ (bottom left), when the number of samples $n_{mc} \to \infty$ (top left) and when Δt is not too large or small (bottom right).

complex 3D fence topology of Figure 5.14 exceeds the limits of what the simplified $\Psi = \vec{0}$ boundary condition is expected to accurately support, leading to somewhat more damped flow from one side to the other. Extending WoS to support the $\Psi = \nabla\phi$ boundary condition of Ando et al. [1] may be one avenue to address this shortcoming.

Convergence: In the limit as Δt goes to zero and the number of samples approaches infinity, our method should converge to the correct solution (up to the accuracy limited by cache, when applicable). Figure 5.10 qualitatively demonstrates that our method indeed gives a result that is consistent with existing techniques.

We opted to evaluate the convergence of our method using the steady-state inviscid Taylor-Green vortex flow [186], for which a closed-form solution is known. Figure 5.15 shows log-scale plots of the root mean squared error (RMSE) against the number of BS samples n_{mc}, number of cache cells $n_{\text{c}} = n_{\text{x}}^2$ and time step interval Δt. All errors were computed at the physical times $t = 0.25, 0.5, 1$ using IS of the vorticity, bilinear interpolation, RK4 advection, without CV. We also let $\Delta t = 2^{-6}$, $n_{\text{x}} = 2^{10}$ and $n_{\text{mc}} = 2^{10}$ whenever they are fixed. A well-known fact about MC methods (Section 3.1.3.1) is that their error diminishes in inverse proportion to the square root of the number of samples, i.e., $\mathcal{O}(1/\sqrt{n_{\text{mc}}})$ [160]. We confirmed this behavior in our experiments. Similarly, we observed $\mathcal{O}(1/\sqrt{n_{\text{c}}})$ which is reasonable given our use of bilinear interpolation. These trends are observed only when the other parameters are good enough to have negligible error contributions. Finally, we observed a convergence order of approximately $\mathcal{O}(\Delta t^{0.85})$. However, convergence is lost whenever Δt gets too small. This regime change is in line with the use of a semi-Lagrangian advection scheme [202]. Intuitively, the error stops decreasing when the interpolation error starts dominating the advection error. Since more steps are needed to reach the same physical time, the error increases as Δt gets smaller.

3D Simulations: We demonstrate our method in 3D, with and without obstacles (Figure 5.12) using CUDA parallel GPU implementations. The leapfrogging simulation uses a uniform cache resolution of 256^3, trilinear interpolation, the CV method enabled, $\Delta t = 0.1$, $n_{\text{mc}} = 128$, and $\nu = 0$. The initial velocity field is estimated with $n_{\text{mc}} = 16384$, IS the initial vorticity field. The simulation time is roughly 3.5 seconds per step on a

machine with two NVIDIA Tesla P100 GPUs at these settings. The Stanford bunny obstacle with slip boundaries uses a uniform cache resolution of 128^3, trilinear interpolation, IS the harmonic Green's function, $\Delta t = 0.1$, 128 WoS paths with maximum 8 steps, 1024 samples for the volume integral evaluation (Eq. 14 in the paper by Sawhney and Crane [165]), and $\nu = 0$. Simulation times are roughly 70 seconds for the triangle mesh boundary and 75 seconds for the triangle soup boundary per step on a machine with four NVIDIA Tesla V100 GPUs using BVH for closest distance computation. We believe this small difference of runtimes (additional 5 seconds for triangle soup) is because of the additional cost for the closest distance evaluation.

We perform no low-level simulator optimization, focusing instead on the method's feasibility; GPU memory access optimizations and faster BVH traversal implementations can be explored.

Control Variates: The method of CV as described in Section 5.3.4 can greatly reduce the variance of simulation. Figure 5.11 shows the effect of our CV. All three simulations use a cache resolution of 128^3, trilinear interpolation, $\Delta t = 0.1$, and $\nu = 0$. The reference simulation (a) uses IS according to the BS kernel with $n_{\text{mc}} = 2048$. The simulation with CV (b) uses the CV strategy in Section 5.3.4 with $n_{\text{mc}} = 32$, with the initial velocity field estimated with $n_{\text{mc}} = 16384$ using IS according to the initial magnitude of the vorticity field. The simulation without CV (c) uses IS according to the magnitude of the vorticity field with the same $n_{\text{mc}} = 32$. We can observe that even with as few as 32 samples per query point, the CV approach produces a result very close to the reference while simply importance sampling the vorticity field leads to large deviations in the smoke motion due to the extensive noise in the estimated velocity field.

Parameter Setting: To better understand the impact of the parameters of our solver on simulation output and performance, we illustrate the effects of the various parameters in Figure 5.9. The first column is a reference simulation using our method with reasonable

high-quality parameter settings: a uniform cache resolution of 512×512, IS based on the vorticity field, bilinear interpolation, $\Delta t = 0.05$ seconds, $n_{\text{mc}} = 1024$ samples, and $\nu = 0$. All other columns modify a single parameter *in a manner that purposefully degrades simulation quality*. The results thus show how changing each parameter can degrade the solution.

Due to MC's stochastic nature, nearest neighbor interpolation produces noisier (i.e., higher variance) results that compound over time. Since bilinear interpolation yields smoother results and is nearly free on, e.g., modern GPU architectures, it is the obvious choice. We also compare two non trivial MC IS functions to reduce the variance of the estimator and, as expected, IS from the vorticity field yields sharper and more accurate results than sampling according to the BS kernel (see Section 5.3.4) since the latter decreases slowly and omits information of the current simulation state. The discrepancy here is particularly strong when the vorticity is sparsely distributed across the domain. While our advection scheme is unconditionally stable, using time steps that are too large or a cache resolution that is too low results in inevitable loss of detail in high variation regions. Finally, since standard MC error decreases at a rate of $\mathcal{O}(1/\sqrt{n_{\text{mc}}})$, the number of samples required for high quality results can be high; when we reduce the sample count by a factor of four, we note a significant loss of details.

Chapter 6

Discussion

We will delve into the implications of the methods presented in Chapter 4 and Chapter 5. In particular, we will outline some limitations and potential future research directions associated with these methods. Additionally, we will introduce several ideas that we explored alongside these projects but ultimately did not result in publication.

6.1 Delayed Rejection Metropolis Light Transport

In this project, we introduced a two-stage mutation strategy based on the delayed rejection (DR) framework, which is flexible and capable of balancing specialized mutation strategies within tight computational budgets. While delayed rejection Metropolis light transport (DRMLT) performs well in a variety of scenarios, there are still some limitations in its current state. We discuss them below, in addition to several interesting and promising research directions. This section is inspired by the many discussions that we presented in our previous publication [157].

6.1.1 Portability to Path Space Metropolis Light Transport

There are several challenges in porting the original DR algorithm from Tierney and Mira [188] to Veach and Guibas [193] Metropolis light transport (MLT). As path space perturbation strategies are mostly complementary and designed for paths with specific structures, choosing perturbation strategies for the two stages that are both *general* and *suitable* is hardly feasible. As such, a complex decision tree based on the possible path types should be established, further obfuscating the mutation routine. We can apply our orbital mutation directly to simple perturbations (e.g., lens and caustics) but more intricate mutation strategies (e.g., manifold exploration [78]) need to be treated separately. This could potentially be accomplished through inverse mappings [144, 147, 14], transforming paths to their primary sample space (PSS) representations. We leave this as future work.

Extra care is also needed when adapting Green and Mira's Green and Mira [58] algorithm to path space MLT. Indeed, this generalized version of DR requires tracing an intermediate light path $\boldsymbol{y}^* = \boldsymbol{z} - \delta\boldsymbol{y}$. Naïvely adding or transforming vertices will almost always result in paths that escape the scene manifold $\mathcal{M}$. One way of alleviating this issue is to project each vertex back onto the scene to obtain a new intermediate path $\boldsymbol{y}' = \mathsf{proj}_{\mathcal{M}}(\boldsymbol{y}^*)$. Accounting for this projection in Equations (4.8–4.9) to retain reversibility, however, may be difficult.

6.1.2 Number of Stages

Our approach only attempts two stages before conceding rejection and advancing time. While DRMLT can be generalized to m stages by enforcing detailed balance separately at each stage $i \leq m$, some problems remain. This construction gives rise to a product of rejection probabilities $\prod_i Q_i[1 - \alpha_i]$ that can magnify the vanishing acceptance phenomenon in subsequent stages [124]. Moreover, the number of reversibility

constraints for these states grows geometrically with the number of stages, incurring a non-negligible additional cost. As noted by Green and Mira [58], moving to three or more stages is typically not worthwhile, motivating our two-stage mechanism.

6.1.3 Choice of Proposals

One challenge when applying DRMLT lies in devising a sufficiently diverse sequence of proposals. This requires understanding the scale at which mutations operate and crafting a second stage mutation strategy that can capitalize on the shortcomings of the first. Finding innovative combinations that naturally identify these flaws is non-trivial. Also, in some cases, a grid search for "optimal" kernel spread and/or mixture weights could yield performance similar to our bold-then-timid approach. By design, DRMLT precisely avoids such tedious trial-and-error experiments and circumvents scene-dependent parameter optimization.

6.1.4 Differential Geometric Mutations

Given the growing ubiquity of differentiable rendering [140, 102], our approach offers an attractive tool for amortizing the per-sample cost incurred by automatic differentiation. There are other Markov chain Monte Carlo (MCMC) samplers (e.g., Metropolis adjusted Langevin algorithm (MALA) [108] and its Riemannian manifold variant manifold Metropolis adjusted Langevin algorithm (MMALA) [54]) that exploit the geometry of the target distribution to improve exploration. A technique based on that was recently introduced to MLT by Luan et al. [109]. It is worth revisiting alternatives that were promptly dismissed for light transport simulation due to their seemingly high cost. Applying DR to Hessian Hamiltonian Monte Carlo (H2MC) was the first step in this direction, and investigating the performance of other differentiable mutations at the second stage is a promising avenue of future work.

6.1.5 Improved Large Steps

DR can be applied to the large step mutation, similar to Trias et al. [189]. When a large step falls in the neighborhood of a target mode but is rejected by Metropolis–Hastings (MH), DR could be applied to reach this mode with successively smaller steps to climb up the density hill. This effectively explores the new region of parameter space until a new state is finally accepted or a stopping condition is met. Identifying when an initial large step must be followed by additional small steps remains unclear, but doing so on a stochastic basis may foster global exploration.

6.1.6 Delayed Rejection Adaptive Metropolis

Combining adaptive MCMC with DR in a method called delayed rejection adaptive Metropolis (DRAM), as suggested by Haario et al. [64], is a natural direction to explore. This method relies on constructing proposals by fitting the covariance of the distribution at different scales. One may (perhaps optimistically) expect to achieve similar performance as Li et al. [101] here, however, without the need for automatic differentiation. DRAM gives rise to a correlated transition kernel and is built atop the original DR algorithm, and so more investigation is required to address any potentially negative effects on the acceptance rates.

6.2 A Monte Carlo Method for Fluid Simulation

Our work is the first step towards a new family of fluid solvers based on Monte Carlo (MC) methods. We demonstrated the potential of our MC-based fluid solvers to generate high quality and accurate results, such as when confronted with complex and uncleaned geometry, but there remains much to be explored. A better understanding on how we can best exploit the unique properties, and new trade-offs of noise and computation time within fluid animation to name a few general directions. Echoing Sawhney and Crane

[165], open directions include geometric robustness and flexibility, parallelism, adaptivity and output sensitivity, reduced (or eliminated) dependence on mesh discretization, and further reduction of variance. Below we discuss limitations and propose avenues for future work. This section is inspired by the many discussions that we presented in our previous publication [158].

6.2.1 Control Variate

The control variates we used in Equation (5.23) is not the general formula for control variates. The general form uses a mixture between the estimator and control function with a control parameter. It would be interesting to study how one could introduce an optimal control parameter in our setting to further reduce the estimator variance.

6.2.2 Bias

Currently, our method has two sources of bias—the first one coming from the cache interpolation error and the other from the semi-Lagrangian advection—resulting in energy loss as time advances. The first source, e.g., the cache interpolation bias, is not inherent to our fundamental formulation. One way this could be resolved, without exponential cost, is through the use of a shared particle cache (and sampled locations for Monte Carlo integration) if one allows recursion to go back to the initial conditions. However, the second source, e.g., the advection bias, is intrinsic to our advection process and solving this issue remains an open question. It comes from the fact that the nonlinearity of our advection scheme (Equation (5.3)) does not commute with the MC estimator, much like how naive transmittance estimation through ray marching is biased [141]. Further down the road, it would be interesting to explore whether the work of Misso et al. [125] can be used in our context to generate an unbiased estimator.

6.2.3 Semi-Lagrangian Dissipation

For tracing semi-Lagrangian trajectories, we used either forward Euler or Runge–Kutta 4th order (RK4), and as expected observed somewhat improved results from the latter option. The dissipation that remains comes from interpolating from the cache, similar to grid-based methods; using higher order interpolation therefore ought to result in less dissipation. It would be interesting to look at incorporating more advanced advection or time-splitting schemes, such as MacCormack, BDF2, advection-reflection, IVOCK, etc., to further lower the dissipation.

6.2.4 Boundary Conditions

Investigating a broader set of boundary conditions, such as no-slip—the main source of vorticity in viscous flows—and mixed boundaries is another interesting direction.

While some approaches existed when we submitted this publication [113, 176], effectively handling Neumann and mixed boundaries in Walk-on-Spheres (WoS) was still an open problem, as discussed, for example, by Sawhney et al. [167]. However, Sawhney et al. [168] recently created the Walk-on-Stars algorithm to efficiently handle Neumann boundary condition. It would be quite interesting to see how this new framework could be integrated to ours.

In general, our free-slip boundary condition framework is only effective for a single closed and connected boundary; otherwise, one needs to know the difference in stream function isovalues between the separate boundaries. Similarly, a 3D vector potential is further complicated by its non-trivial null spaces; however, successfully generalizing to multiple disjoint objects could benefit from MC's ability to handle far more complex geometry, as explored by Sawhney and Crane [165]. Exploring applications of the Kelvin transform [136] in an MC fluid context would enable efficient simulations on infinite domains, with or without obstacles.

Figure 6.1: Uniform and adaptive caches. Fluid simulation results using a uniform cache (left) and an adaptive cache (middle and right) using one-twentieth memory footprint that of the uniform one. Slowly varying regions are sparsely cached while dense sampling occurs in regions exhibiting steep or discontinuous variations.

Besides inflow and outflow boundaries, we assume no external forcing or sourcing, but a generalization of the Feynman-Kac theorem can be used to incorporate these effects [22]. Extensions to liquids are more challenging, as they require new free surface boundary conditions [110] and a deforming surface representation.

6.2.5 Advanced Caching Methods

We have presented how adopting a uniform grid cache avoids exponential cost and makes the method feasible in practice. However, our framework is not limited to a uniform grid and we can consider several different alternatives, each with associated trade-offs. We believe that adaptive caching methods or a data structure similar to particle-based/hybrid methods could all be viable alternatives. A detailed study is beyond the scope of our initial exploration of MC for fluid animation. Nevertheless, we comment on two possible alternative implementations with preliminary examples.

The first alternative is an adaptive grid cache, replacing the uniform grid with an adaptive tree structure to exploit sparsity. Due to our method's pointwise estimate feature, we can reduce the storage cost for caching *without* affecting the MC estimator

Figure 6.2: Particles cache. An alternative implementation with Lagrangian particle cache (right) yields results consistent with our baseline uniform grid (left).

itself; other methods typically require modifications to their numerical algorithms such as discrete gradient computation, for example. Figure 6.1 illustrates our adaptive cache prototype result. It exhibits a large reduction in memory, although this prototype does not yet offer a significant speed gain.

Another exciting alternative uses a set of scattered (unstructured) cache points. Such points could be formed from Lagrangian particles, e.g., particles advected along the simulation, Eulerian, e.g., that are not advected, or even both. Lagrangian point caches can be constructed by importance sampling (IS) the particles with respect to the initial vorticity strength, ensuring that particles are distributed proportionally to the vorticity field at all times. Using this type of cache makes our MC approach applicable also to a *particle-based solver*, albeit based on an entirely different mathematical framework than the existing particle-based solvers. Figure 6.2 shows that this method gives a consistent result with our grid cache-based method.

Caches are also used in rendering to accelerate the computation of multiple bounces of light [198, 93, 80]. Since our equations have a similar structure as a recursive equation in rendering, more careful allocation of cache points and reconstruction in our problem can also be beneficial. Those prior work in rendering, however, would not be directly applicable to our method since fluid dynamics and light transport have different characteristics in terms of estimation errors (e.g., the split-sphere model in radiance caching [198] has no clear role in our framework).

Lastly, Miller et al. [123] recently created a caching strategy based on the boundary element method to accelerate WoS computation, although after the publication of our work. It would be quite interesting to see how this new framework could be integrated to ours for more efficient caching. It also has the advantage to handle their Walk-on-Stars algorithm [168], which is an extension of the WoS algorithm for Neumann and mixed boundary conditions.

6.2.6 Alternatives to Caching

Our cache-based implementation, and any alternative caching methods discussed in the previous section, result in additional bias beyond that introduced by time discretization. This issue motivates a desire to explore and devise alternative solutions for resolving the exponential cost of our base recursive formulation. This exponential cost, in its essence, is similar to the problem of having an exponential cost in path tracing [83] if we were to split a path at each bounce (i.e., exponential to the number of bounces). Path tracing avoids this exponential growth by tracing along only *one* direction per bounce. While we have empirically found that this approach is not applicable in our setting, some further study of MC methods may avoid this exponential cost.

Quantum methods for numerical integration [174] may be one potential avenue due to the exponential nature of computation via quantum bits. Russian roulette [4], an unbiased method to terminate recursive MC processes according to their expected contribution, can be applied to "early-out" fluid trajectories that contribute negligibly to the final dynamics. Similarly, path (resp. trajectory) splitting [196] can be used to adaptively refine our sampling.

6.2.7 Variance Reduction

While the CV approach we presented for the Biot–Savart (BS) case is quite effective, we have only begun to explore the broader landscape of variance reduction methods in MC. Designing an efficient sampling strategy in WoS would be crucial to make the method more practical in the presence of solid boundaries. Compared to the BS case, the design space to explore for effective importance functions for WoS is even more flexible. For the boundary integration in WoS, it would be interesting to investigate how directional importance functions, such as the von-Mises Fisher or circular Cauchy distributions [48], could be leveraged. Drawing inspiration from the field of light transport, we can employ multiple importance sampling (MIS) to combine *several potentially good* probability density function (PDF) into more robust and sample efficient strategies. Reusing the sample WoS paths within each iteration or even over consecutive time steps similarly to ReSTIR [15] could also be an interesting direction. Since our approach is based on MC, it is compatible to MCMC methods [21] and significant variance reduction akin to MLT [194] might be possible in our MC fluid solver. A concurrent work on the bidirectional WoS method [154] could be combined with our method.

6.2.8 Tighter Coupling with Rendering

Perhaps most excitingly, the similarities of our method to MC-based light transport simulations suggest that a tighter integration between fluid and light transport simulations may prove fruitful. For example, since our fluid simulation generates errors that manifest as noise, and since possibly only a subset of the domain is rendered from any individual camera setting, one may be able to adapt the simulation sampling rate to account for the precision needed by the rendering algorithm. Last-mile denoisers of rendered images [24, 84], further diversify this design space and trade-offs therein. For

example, one might be able to design a denoiser which removes noise from *both* rendering and simulation (via our method) in a coupled manner.

Relatedly, heterogeneous volumetric rendering algorithms [141] build acceleration structures and variance reduction techniques to accelerate their convergence, and these structures can inform (and be informed) by MC estimates of the underlying fluid dynamics. For example, one could imagine a hybrid approach in which a coarse grid-based simulation is used to guide the rendering process and high resolution local resimulations are performed on the fly using our approach when the renderer requires higher-resolution fluid density data.

6.3 Unpublished Supporting Work

In addition to the two main projects and the publication mentioned in Chapter 1.2, we explored several unpublished projects and ideas. We will now take a moment to discuss some of them, although these are just the tip of the iceberg.

6.3.1 Quasi-Newton Metropolis Adjusted Langevin Light Transport

Two primary factors hinder the widespread adoption of differentiable-based light transport algorithms, such as the one proposed by Li et al. [101]. Firstly, evaluating the derivative of the target density is challenging, requiring additional frameworks like automatic differentiation. Secondly, assessing differential quantities, particularly the Hessian, is computationally demanding even when using these frameworks. This latter issue inspired the method presented in Chapter 4 and prompted us to explore ways to eliminate the need for Hessian computation altogether.

To achieve this, we examined the combination of the MMALA [54] with a quasi-Newton Markov chain Monte Carlo (QN-MCMC) [208] Hessian approximation. This exploration coincided with the publication of a similar project by Luan et al. [109],

who proposed using MALA with Adam preconditioning and a caching strategy to improve large step mutations. Given the similarities between our project and theirs, and the conclusions they reached surrounding the use of QN-MCMC in this context, we decided to shift our focus to other projects instead. Nevertheless, this exploration gave us a lot of insights and future work ideas.

For instance, consider gradient domain MLT [99] where the target density is usually highly anisotropic due to the inclusion of a discrete gradient component. One idea would be to apply the MMALA technique in conjunction with the gradient domain's shift mapping operator to better explore the target density. We strongly believe that this promising direction offers a rich area for future exploration and could lead to significant advancements in the field of gradient domain.

6.3.2 Kernel Adaptive Metropolis Light Transport

In this project, our objective was to entirely eliminate the need for differentiable information. We drew inspiration from the work of Sejdinovic et al. [171], which embeds Markov chain trajectories into a reproducing kernel Hilbert space (RKHS), allowing the feature space covariance of the samples to guide the selection of the proposal. This method produces a Gaussian proposal whose covariance relies on both the current sample's position and the local statistical properties of the target distribution. However, the approach necessitates an initial pass to generate samples, which are then utilized to construct the proposal through the RKHS embedding. Unfortunately, we discovered that the complexity of light transport made it challenging to acquire enough samples in all areas to reconstruct effective proposals, especially when dealing with highly glossy interactions. We also tried using a limited window of past samples to address this issue, but managing the vanishing adaptation criterion proved too difficult. This situation resembled a chicken-or-egg dilemma: We needed quality samples to create a suitable proposal, yet we also required an effective proposal to generate samples. With that in

mind, we are optimistic that devising a solution to this challenge could be a valuable direction for future research. Such an endeavor would aim to develop a method that rivals the effectiveness of differential-based approaches, but without their associated costs and implementation complexities.

6.3.3 Delayed Acceptance Metropolis Light Transport

Similar to DR, we believed that delayed acceptance (DA) [27, 173] could be employed to combine a cost-effective approximation of the target density with a two-stage version of MH. DA tests samples against the surrogate target, quickly dismissing those unlikely to be accepted by the true target. A second stage then uses an acceptance probability to resolve the discrepancy between the approximation and the target, ensuring convergence. Since evaluations of path throughput are generally the bottleneck of MLT-based methods, a successful application of this method could lead to significant speedups. We attempted two such strategies.

Path Reconnection: One of the most costly components of the MLT algorithm using bidirectional path tracing (BDPT) as path sampler is vertex reconnection during the computation of the MIS strategy. We initially considered employing the result without the visibility term as a surrogate target. However, it appears that most rejected paths originate from the visibility term, which means that the surrogate density does not have the opportunity to filter them out. Consequently, this naive approach failed to accelerate computation in this specific case. Finding a way to make the surrogate better through an approximation of the visibility term would make for an interesting research direction.

Russian Roulette: During path tracing, the Russian roulette algorithm randomly terminates path construction with a probability inversely proportional to the throughput [152]. We attempted to use an aggressive termination probability at the first stage to

minimize the cost of tracing paths when their contributions are too low. Then, if this path is accepted during this first stage, the path throughput is corrected and extended with a milder termination probability during the second stage. This approach did not yield positive results as paths that have interesting contributions get rarer and rarer as we go up in dimensions. Finding ways to overcome this problem would be an interesting topic for future exploration.

From these two experiments, we concluded that the surrogate function must be an excellent approximation of the true target function in order to be effective. Coming up with one that is also very inexpensive to evaluate is highly challenging to achieve in a domain as complex as light transport, and for this reason, we did not pursue our inquiries further. We do believe however thatthis topic merits further exploration, particularly in light of the recent advancements in neural-based approximations within the field of light transport. Additionally, there are still a few other ideas relying on delayed acceptance that might be worth exploring, such as using a surrogate based on approximate ray tracing through spatio-directional data structures [132] or employing coarse volumetric approximations in volumetric rendering [95].

Chapter 7

Conclusion

We now arrive at the culmination of this thesis. To begin, a comprehensive summary will encapsulate the key findings and ideas presented throughout this work. This will be followed by concluding remarks, where we will reflect on the broader implications of our findings, their relevance to the existing body of knowledge, and potential avenues for future research.

7.1 Summary

To summarize, our goal in this thesis was to develop flexible Monte Carlo (MC) methods for computer graphics applications, specifically for physically based rendering and physically based fluid animation. To achieve this, we presented two methods, each targeting advancement within their respective fields. Below is a summary of the content in this monograph:

In Chapter 2, we offered a comprehensive review of relevant work surrounding our methods.

In Chapter 3, we supplemented the previous chapter with a detailed exposition of the background knowledge necessary to understand the methods presented in subsequent chapters.

Our first contribution, detailed in Chapter 4, addressed the challenge of designing efficient and effective mutation strategies for Metropolis light transport. We introduced a two-stage mutation strategy based on the delayed rejection (DR) framework, which provided a *safety net* to enhance local state space exploitation without compromising global exploration. We demonstrated that a naïve application of the original DR algorithm to light transport simulation could result in poor acceptance rates and developed a novel conditional mutation strategy for the second stage. Our resulting algorithm, delayed rejection Metropolis light transport (DRMLT), is straightforward to implement in existing rendering systems and can be applied to any primary sample space-based algorithm. We showcased various applications of DRMLT, which automatically balance local exploration and computational efficiency across numerous complex light transport scenarios.

Our second contribution, presented in Chapter 5, proposed a MC framework for solving fluid motion that is flexible and robust to geometric complexity. Our solver generates stochastic pointwise solutions to the incompressible Navier–Stokes equations based on an application of the Feynman–Kac formula to the vorticity transport equation. We developed a simple, parallelizable implementation using a cache to counter the exponential cost of a naïve treatment of our recursive estimator formulation. We further extended our method to handle boundary conditions for free-slip moving solids, inflows, and outflows, employing a stream function formulation and the Walk-on-Spheres algorithm to address the associated Poisson problem. This enables simulations with complex boundary shapes with minimal effort. We applied three variance reduction techniques, namely importance sampling, antithetic sampling, and control variates, to accelerate computation. As the first such numerical solver and a significant departure from standard approaches, our method highlights the distinctive characteristics of MC fluid animation, such as its adaptability to geometric complexity.

Finally, in Chapter 6 we provided an extensive discussion of our findings, including limitations and future research avenues. We also included information about some unpublished projects and ideas explored during this Ph.D. thesis.

7.2 Concluding Remarks

We believe that these two contributions represent promising steps toward advancements in their respective domains. An intriguing direction for future research would be to explore the possibility of rendering and simulating fluid using a single unified MC method, leveraging our newly proposed frameworks. Indeed, since our new MC method for fluid simulations provides a means of obtaining pointwise approximations of fluid motions, it could be integrated to MC rendering algorithms such as our DRMLT framework, enabling path tracing and approximation of fluid motion simultaneously. More specifically, this could be applied to a *simulate only what you see* strategy, which would be beneficial in situations where simulating a vast domain, such as an ocean, is inefficient when only a portion is visible at any given moment. Of course, numerous complex issues must be addressed before achieving this level of integration, but we are optimistic that our innovative methods have set the stage for such advancements.

Appendix A

Measure Theory

This appendix exposes the basic concepts of measure theory. Our goal is to offer the reader with intuitions and insights, rather than writing a measure theory textbook. Therefore, we will keep most of the discussion at a high level and minimize the level of rigor, as it can become tedious. If you are interested in studying measure theoretic probability theory and stochastic processes in more details, the following textbooks are excellent resources to consult [142, 8, 162, 11].

In a nutshell, measure theory is a branch of mathematics that deals with the generalization of the concepts of length, area, and volume in more abstract settings. It provides a rigorous foundation for understanding how to assign sizes, volumes, or probabilities to mathematical objects, and it plays a crucial role in various areas of mathematics, including probability theory, analysis, and geometry.

A.1 Measure Space

A measure space is a mathematical structure that provides a rigorous foundation for the study of measure theory. In a nutshell, a measure space consists of three components: a *sample space* $\boldsymbol{S}$ which is a non-empty set representing the universe of all possible elements under consideration; a *sigma-algebra* $\boldsymbol{\Sigma}$ which is a collection of subsets of $\boldsymbol{S}$ that we want

to measure and that contains the empty set, is closed under taking complements, and is closed under countable unions or intersections; and a *measure* μ which is a function that assigns non-negative measured values to each element of $\boldsymbol{\Sigma}$.

More formally, consider a ***sample space*** $\boldsymbol{S} \neq \varnothing$. A ***sigma-algebra*** $\boldsymbol{\Sigma}$ over the set $\boldsymbol{S}$, also written σ-algebra, is a collection of subsets of $\boldsymbol{S}$ that satisfies the following properties:

(I) The empty set $\varnothing$ and the whole set $\boldsymbol{S}$ are both in $\boldsymbol{\Sigma}$: $\varnothing, \boldsymbol{S} \in \boldsymbol{\Sigma}$.

(II) $\boldsymbol{\Sigma}$ is closed under complements: if $A \in \boldsymbol{\Sigma}$, then $A^c \in \boldsymbol{\Sigma}$.

(III) $\boldsymbol{\Sigma}$ is closed under countable unions: if $A_1, A_2, \ldots \in \boldsymbol{\Sigma}$, then $\bigcup_{i=1}^{\infty} A_i \in \boldsymbol{\Sigma}$.

The pair $(\boldsymbol{S}, \boldsymbol{\Sigma})$ is called a ***measurable space*** and elements of $\boldsymbol{\Sigma}$ are called ***measurable sets***. A ***measure*** is a function $\mu : \boldsymbol{\Sigma} \to \mathbb{R}_{\geq 0}$ which satisfies the following properties:

(I) $\mu(A) \geq 0$ for all $A \in \boldsymbol{\Sigma}$.

(II) $\mu(\varnothing) = 0$.

(III) For any countable collection A_n of ***pairwise disjoint sets*** in $\boldsymbol{\Sigma}$, meaning $A_i \cap A_j = \varnothing$ for $i \neq j$, we have:

$$\mu\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} \mu(A_n)\,. \tag{A.1}$$

The triple $(\boldsymbol{S}, \boldsymbol{\Sigma}, \mu)$ is called a ***measure space***. A property P is said to hold ***almost everywhere*** on $\boldsymbol{S}$ if there exists a subset $A \in \boldsymbol{\Sigma}$ with $\mu(A) = 0$ such that the property P holds for all $\omega \in \boldsymbol{S}$ A.

A.2 Measurable Function

Now we need a function that preserves the structure of measurability between two measurable spaces. Given two measurable spaces $(\boldsymbol{S}_1, \boldsymbol{\Sigma}_1)$ and $(\boldsymbol{S}_2, \boldsymbol{\Sigma}_2)$ a function $f : \boldsymbol{S}_1 \to \boldsymbol{S}_2$ is called a $\boldsymbol{\Sigma}_1$***-measurable function*** if, for any measurable set $B \in \boldsymbol{\Sigma}_2$, the pre-image of B under the function f is a measurable set in $\boldsymbol{\Sigma}_1$, i.e., if $f^{-1}(B) \in \boldsymbol{\Sigma}_2$ for all $B \in \boldsymbol{\Sigma}_2$. When clear from the context, we will simply call f a measurable function.

A.3 Lebesgue Integration

To define the Lebesgue integral, we need a measurable space $(\boldsymbol{S}, \boldsymbol{\Sigma})$ and a measure μ defined on this space. Let $f : \boldsymbol{S} \to \mathbb{R}$ be a non-negative, $\boldsymbol{\Sigma}$-measurable function.

The ***Lebesgue integral*** of f with respect to the measure μ over a set $A \in \boldsymbol{\Sigma}$, denoted by $\int_A f \, \mathrm{d}\mu$, is defined as the supremum of the integrals of ***simple functions*** g, e.g., finite linear combinations of characteristic functions of measurable sets that are less than or equal to f:

$$\int_A f \, \mathrm{d}\mu = \sup \left\{ \int_A g \, \mathrm{d}\mu \,|\, g \text{ is simple}, 0 \leq g \leq f \right\} . \tag{A.2}$$

Here, the ***integral of a simple function*** g with respect to μ is given by the sum of the product of the measure of each measurable set $A_i \in \boldsymbol{\Sigma}$ and the corresponding value g_i:

$$\int_A g \, \mathrm{d}\mu = \sum_k g_k \, \mu(A_k \cap A) \, . \tag{A.3}$$

The Lebesgue integral can be extended to include not only non-negative functions but also real-valued functions by decomposing them into their positive and negative parts. If f is a real-valued, measurable function, we can write $f = f^+ - f^-$, where $f^+ = \max(f, 0)$ and $f^- = \max(-f, 0)$ are non-negative, measurable functions. If at least one of the Lebesgue integrals of f^+ and f^- is finite, then the ***signed Lebesgue integral*** of f is defined as:

$$\int_A f \, \mathrm{d}\mu = \int_A f^+ \, \mathrm{d}\mu + \int_A f^- \, \mathrm{d}\mu \, . \tag{A.4}$$

The Lebesgue integral is a generalization of the Riemann integral that is more robust to limits and ill-behaved sets. In addition, it is not restricted to the usual length measure. For this reason, it offers a uniform notation for integrating against any measure and generalizes the concept of sum and integral.

A.4 Radon–Nikodym Theorem

Often, we need to change the measure over a measurable space. To do so we need to define the absolute continuity of measure and the Radon–Nikodym theorem. Given a measurable space $(\boldsymbol{S}, \boldsymbol{\Sigma})$, let μ and ν be two measures defined on this space. The measure ν is said to be ***absolutely continuous*** with respect to μ (denoted by $\nu \ll \mu$) if, for any measurable set $A \in \boldsymbol{\Sigma}$, the condition $\mu(A) = 0$ implies $\nu(A) = 0$. The ***Radon–Nikodym theorem*** states that if $\nu \ll \mu$, then there exists a non-negative measurable function f, called the ***Radon–Nikodym derivative***, such that for any measurable set $A \in \boldsymbol{\Sigma}$:

$$\nu(A) = \int_A f \, \mathrm{d}\mu \,. \tag{A.5}$$

This function is often denoted $f = \frac{\mathrm{d}\nu}{\mathrm{d}\mu}$ to reflect the calculus notation. It is also often called the ***density of the measure ν with respect to the measure μ***.

A.5 Common Measures

As the name implies, a measure is used to generalize the concept of counting, length, area, volume as well as probability distribution. Here is a quick overview of some of the measures that we will encounter in this thesis.

Probability Measure: A ***probability measure*** $\mathcal{P}$ is a measure that satisfies $\mathcal{P}\boldsymbol{S} = 1$. This means that:

$$\int_{\boldsymbol{S}} \mathrm{d}\mathcal{P}(\boldsymbol{y}) = 1 \,. \tag{A.6}$$

Counting Measure: The ***counting measure***, $\#$, assigns a non-negative integer value to each subset of a countable set:

$$\#(A) = |A| \quad \text{e.g., the number of elements of A} \,. \tag{A.7}$$

It is commonly used to define the concept of probability mass function (PMF) (Section 3.1.1.2), which is utilized to represent the distribution of a discrete random variable (RV). When working with discrete spaces, this measure will be regarded as the reference measure. In terms of integrating a measurable function f with respect to this measure, we have:

$$\int_A f \, \mathrm{d}\# = \sum_{a_i \in A} f(a_i) \,, \tag{A.8}$$

where $A = \{a_i\}$ is a countable set.

Lebesgue Measure: The ***Lebesgue measure***, denoted λ_d, assigns a non-negative real number to subsets of Euclidean spaces $\mathbb{R}^d$, such as the real line, the plane, or higher-dimensional spaces. It is often considered a generalization of the concept of length, area, and volume as it coincides with the standard way to measure these quantities for well-behaved subsets. It is primarily used to define the notion of a probability density function (PDF) (Section 3.1.1.2), which is used to describe the distribution of a continuous RV. Unless stated otherwise, this measure will serve as a reference measure when working in continuous finite spaces.

More concretely, the Lebesgue measure λ is defined on a sigma-algebra $\mathcal{B}^d$ of subsets of $\mathbb{R}^d$, which contains all ***Borel sets*** (the smallest sigma-algebra containing all open sets of $\mathbb{R}^d$). The construction of the Lebesgue measure is a bit painful so we will not go over it in this appendix. The interested reader can refer to one of the references mentioned at the start of the appendix. What is more important are the properties that it satisfies:

- The Lebesgue measure of an interval $(a, b) \in \mathbb{R}$ is given by $\lambda((a, b)) = b - a$, which is the usual length $|(a, b)|$ of the interval.
- If $A = I_1 \times \ldots \times I_d$ is a Cartesian product of intervals, then $\lambda(A) = |I_1| \cdot \ldots \cdot |I_d|$
- **The Lebesgue measure is countably additive:** If $\{A_i \in \mathcal{B}\}$ is a countable collection of pairwise disjoint measurable sets, then $\lambda\left(\bigcup A_i\right) = \sum \lambda(A_i)$.
- **The Lebesgue measure is translation invariant:** For any measurable set $A \in \mathcal{B}^d$ and any vector $\vec{x} \in \mathbb{R}^d$, $\lambda(A + \vec{x}) = \lambda(A)$, where $A + \vec{x}$ denotes the translation of A by $\vec{x}$.

Wiener Measure: The ***Wiener measure***, often denoted as μ_W, is defined on the space of continuous functions $C([0,T],\mathbb{R})$ representing the continuous paths from $[0,T]$ to $\mathbb{R}$ with $0 < T < \infty$. It is used to associate probabilities with continuous-time stochastic processes such as random walks, also called Brownian motions. In the context of random processes, this measure will be considered the reference measure when working in continuous infinite space (function space).

Dirac Measure: Usually denoted $\mu = \delta_\omega(A)$, the ***Dirac measure*** is a probability measure that assigns 1 to A if $\omega \in A$ and 0 otherwise, i.e.,

$$\delta_\omega(A) = \begin{cases} 0 & \text{if } \omega \notin A\,, \\ 1 & \text{if } \omega \in A\,. \end{cases} \tag{A.9}$$

In terms of integrating a measurable function f with respect to this measure, we have

$$\int_{\boldsymbol{S}} f(\boldsymbol{y})\,\mathrm{d}\delta_{\{\boldsymbol{x}\}}(\boldsymbol{y}) = f(\boldsymbol{x})\,. \tag{A.10}$$

It concentrates all the *mass* at the single point $\boldsymbol{x}$ and can be thought of as an *infinitely concentrated* measure. Note that the Dirac measure is not absolutely continuous with respect to the Lebesgue measure when $\boldsymbol{S} \subset \mathbb{R}^d$. For this reason, it does not have a PDF in the traditional sense. However, it can be represented using the ***Dirac delta function*** δ (which is not a true function but rather a distribution) in the context of generalized functions, which can be understood as the (weak) limit of a sequence of bumb functions:

$$\delta(\boldsymbol{x}) = \lim_{b\to\infty} \frac{1}{|b|\sqrt{\pi}} \exp\left[-\left(\frac{\|\boldsymbol{x}\|}{b}\right)^2\right]. \tag{A.11}$$

Loosely speaking, it can be understood as the function:

$$\delta(\boldsymbol{y}-\boldsymbol{x}) \approx \begin{cases} 0 & \text{if } \boldsymbol{x} \neq \boldsymbol{y}\,, \\ \infty & \text{if } \boldsymbol{x} = \boldsymbol{y}\,, \end{cases} \tag{A.12}$$

with constraint:

$$\int_{\boldsymbol{S}} \delta(\boldsymbol{x})\,\mathrm{d}\lambda(\boldsymbol{x}) = 1\,. \tag{A.13}$$

This measure arises in light transport when considering a point light, a pinhole camera and purely specular reflection where only one deterministic point or direction is possible. With some abuse of notation, we can imagine that the area and solid angle measure (Section 3.2.2) can be extended to handle the Dirac delta function in such cases.

Restricted Measure: A ***restricted measure***, also known as an ***induced measure***, is a measure that is defined on a subset of a measurable space while maintaining the properties of the original measure. They can be used to define a measure on a subset of a space or to define conditional probability distributions. More concretely, consider a measure space $(\boldsymbol{S}, \boldsymbol{\Sigma}, \mu)$ and $A \in \boldsymbol{\Sigma}$ a subset of $\boldsymbol{S}$ onto which we want to define a new measure space induced from the original one. We can define a restricted measure space $(\boldsymbol{S}_A, \boldsymbol{\Sigma}_A, \mu_A)$ on the subset A as follows:

- The ***restricted sample space*** is defined as $\boldsymbol{S}_A = A$.
- The ***restricted sigma-algebra*** is defined as the collection of subsets $B \subseteq \boldsymbol{S}_A$ such that $B = A \cap S$ where $S \in \boldsymbol{\Sigma}$.
- The ***restricted measure*** is defined as $\mu_A(B) = \mu(B)$ for all $B \subseteq \boldsymbol{S}_A$.

Pushforward Measure: A pushforward measure describes the transformation of a measure under a measurable function. It is particularly relevant in probability theory, where it is used to describe the distribution of a RV. Suppose we are given two measurable spaces $(\boldsymbol{S}_1, \boldsymbol{\Sigma}_1)$, $(\boldsymbol{S}_2, \boldsymbol{\Sigma}_2)$, and a measurable function $f : \boldsymbol{S}_1 \to \boldsymbol{S}_2$ between

them. Let μ_1 be a measure defined on the sigma-algebra $(\boldsymbol{S}_1, \boldsymbol{\Sigma}_1)$. The ***pushforward measure*** of μ under the function f, denoted by $f_*(\mu)$, is a measure defined on the $(\boldsymbol{S}_2, \boldsymbol{\Sigma}_2)$ given by:

$$f_*(\mu) = \mu\left(f^{-1}(B)\right) \ , \tag{A.14}$$

where all events $B \in \mathcal{F}_2$. In other words, the pushforward measure $f_*(\mu)$ of a measurable set $B \in \mathcal{F}_2$ is the measure μ of the pre-image of B under the function f in the source space $\mathcal{F}_1$.

In particular, it can also be used to push the Lebesgue measure onto a d-dimensional ***(boundary) manifold*** $\mathcal{M}$ by using the underlying homeomorphisms between the open sets of $\mathcal{M}$ and open sets of $\mathbb{R}^d$ (or the ***closed half-space*** $H^d = \{(x_1, \ldots, x_d) \in \mathbb{R}^d \,|\, x_d \geq 0\}$ on boundaries).

Appendix B

Pseudocode for the DRMLT Algorithm

We summarize our delayed rejection Metropolis light transport (DRMLT) approach in Algorithm B.1. Here, the function MUTATE() depends on the application, as shown in Section 4.3. SAMPLEPATH() maps a random number vector to a light path, ℓ denotes the path luminance, and $\xi \sim U[0,1]$. The MUTATE() function for our pairwise orbital mutation is described in Algorithm B.2.

Algorithm B.1: Delayed rejection Metropolis light transport (DRMLT)

$\boldsymbol{y} \leftarrow \text{MUTATE}\,(\boldsymbol{x}, \_, 1)$, $\ell_{\boldsymbol{y}} \leftarrow \text{SAMPLEPATH}\,(\boldsymbol{y})$
$\alpha_1 \leftarrow 1 \wedge [\ell_{\boldsymbol{y}} \,/\, \ell_{\boldsymbol{x}}]$
if $\xi_1 < \alpha_1$ **then** $\boldsymbol{x} \leftarrow \boldsymbol{y}$, $\ell_{\boldsymbol{x}} \leftarrow \ell_{\boldsymbol{y}}$
else
  $\boldsymbol{z} \leftarrow \text{MUTATE}\,(\boldsymbol{x}, \boldsymbol{y}, 2)$, $\ell_{\boldsymbol{z}} \leftarrow \text{SAMPLEPATH}\,(\boldsymbol{z})$
  $\eta \leftarrow \ell_{\boldsymbol{z}} \,/\, \ell_{\boldsymbol{x}}$
  if Green and Mira [58] **then**
    $\Gamma_2 \leftarrow Q_2(\boldsymbol{x} \,|\, \boldsymbol{y}^*, \boldsymbol{z}) \,/\, Q_2(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{x})$
    $\boldsymbol{y}^* \leftarrow \boldsymbol{z} - (\boldsymbol{y} - \boldsymbol{x})$, $\ell_{\boldsymbol{y}^*} \leftarrow \text{SAMPLEPATH}\,(\boldsymbol{y}^*)$
    $\alpha^* \leftarrow 1 \wedge [\ell_{\boldsymbol{y}^*} \,/\, \ell_{\boldsymbol{z}}]$
    $\alpha_2 \leftarrow 1 \wedge [\eta\,\Gamma_2\,(1 - \alpha^*) \,/\, (1 - \alpha_1)]$ ▷ Equation (4.8)
  else if Tierney and Mira [188] **then**
    $\Gamma_1 \leftarrow Q_1(\boldsymbol{y} \,|\, \boldsymbol{z}) \,/\, Q_1(\boldsymbol{y} \,|\, \boldsymbol{x})$
    $\alpha_2 \leftarrow 1 \wedge [\eta\,\Gamma_1\,(1 - \alpha_1) \,/\, (1 - \alpha_1)]$ ▷ Equation (4.2)
  else if [Pairwise Orbital Mutation – Section 4.2.4] **then**
    $\alpha_2 \leftarrow 1 \wedge [[0 \vee (\ell_{\boldsymbol{z}} - \ell_{\boldsymbol{y}})] \,/\, (\ell_{\boldsymbol{x}} - \ell_{\boldsymbol{y}})]$ ▷ Equation (4.7)
  if $\xi_2 < \alpha_2$ **then** $\boldsymbol{x} \leftarrow \boldsymbol{z}$, $\ell_{\boldsymbol{x}} \leftarrow \ell_{\boldsymbol{z}}$
SPLATCONTRIBUTION $(\alpha_1, \alpha_2, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ ▷ Figure 4.8

Algorithm B.2: MUTATE $(\boldsymbol{x}, \boldsymbol{y}, \text{stage})$

while $i < \text{DIM}\,(\boldsymbol{x})$ **do**
  if stage $== 1$ **then**
    $d \sim Q_{\text{Kelemen}}(\epsilon_{\text{min}}, \epsilon_{\text{max}})$ ▷ Equation (4.5)
    $\phi \leftarrow 2\pi\xi$
    $(\boldsymbol{y}_i,\, \boldsymbol{y}_{i+1}) \leftarrow (x_i,\, x_{i+1}) + d\,(\cos\phi,\, \sin\phi)$
  else ▷ Pairwise Orbital Mutation – Section 4.2.4
    $(u_1,\, u_2) \leftarrow (y_i, y_{i+1}) - (x_i, x_{i+1})$
    $\mu \leftarrow \arccos\,(-u_1 \,/\, \|u\|)$
    if $u_2 > 0$ **then** $\mu \leftarrow 2\pi - \mu$
    $\theta \sim Q_{\text{WCauchy}}(\rho)$ ▷ Equation (4.6)
    $\mu' \leftarrow \mu + \theta$
    $(z_i, z_{i+1}) \leftarrow (y_i,\, y_{i+1}) + \|u\|\,(\cos\mu', \sin\mu')$
  $i \leftarrow i + 2$
if stage $== 1$ **return** $\boldsymbol{y}$ **else return** $\boldsymbol{z}$

Appendix C

Pseudocode for the MCFluid Algorithm

We summarize our Monte Carlo Method for Fluid Simulation (MCFluid) approach in Algorithm C.1.

Algorithm C.1: VOR $(t, \boldsymbol{x})$

Input: Time and position of interest t and $\boldsymbol{x}$
Output: Vorticity at $(t, \boldsymbol{x})$
Data: I is the Identity matrix

if $t == 0$ **then**
  return INITVOR$(\boldsymbol{x})$ `// Initial condition`
$\vec{x}_a \leftarrow \boldsymbol{x} - \Delta t\, \text{COMPVEL}(t - \Delta t, \boldsymbol{x})$ `// Advection Equation` (5.5) `or Equation` (5.8)
$\vec{\omega} \leftarrow \vec{0}$
for $i \leftarrow 1$ ***to*** n_b **do**
  $\vec{x}_i \sim \mathcal{N}(\boldsymbol{x}_a, \sqrt{2\nu\Delta t}\, I)$ `// Diffusion`
  $(\mathcal{D}_{\vec{v}}\vec{\omega})_i \leftarrow$ STRETCH$(t - \Delta t, \boldsymbol{x}_i)$ `// Stretching Equation` (5.21)
  $\vec{\omega} \leftarrow \vec{\omega} + [\text{VOR}\,(t - \Delta t, \boldsymbol{x}_i) + \Delta t(\mathcal{D}_{\vec{v}}\vec{\omega})_i]$
$\vec{\omega} \leftarrow \vec{\omega}/n_d$
return $\vec{\omega}$

Bibliography

[1] Ryoichi Ando, Nils Thuerey, and Chris Wojtan. A stream function solver for liquid simulations. *ACM Transactions on Graphics (TOG)*, 34(4):1–9, 2015.

[2] Christophe Andrieu and Johannes Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373, 2008. doi: 10.1007/s11222-008-9110-y.

[3] Alexis Angelidis and Fabrice Neyret. Simulation of smoke based on vortex filament primitives. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 87–96, 2005.

[4] James Richard Arvo and David Kirk. Particle transport and image synthesis. *Computer Graphics (Proceedings of SIGGRAPH)*, 24(4):63–66, September 1990. ISSN 0097-8930. doi: 10/dtp6gd.

[5] Michael Ashikhmin, Simon Premože, Peter Shirley, and Brian Smits. A variance analysis of the Metropolis light transport algorithm. *Computers & Graphics*, 25(2): 287–294, 2001. doi: 10.1016/S0097-8493(00)00131-X.

[6] Yves F. Atchade. An adaptive version for the Metropolis adjusted Langevin algorithm with a truncated drift. *Methodology and Computing in Applied Probability*, 8(2):235–254, 2006. doi: 10.1007/s11009-006-8550-0.

[7] Vinicius C Azevedo, Christopher Batty, and Manuel M Oliveira. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

[8] Robert G Bartle. *The elements of integration and Lebesgue measure*. John Wiley & Sons, 2014.

[9] Denis R Bell. *The malliavin calculus*. Courier Corporation, 2012.

[10] Jan Bender and Dan Koschier. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 147–155, 2015.

[11] Patrick Billingsley. *Probability and measure*. John Wiley & Sons, 2008.

[12] Benedikt Bitterli and Wojciech Jarosz. Selectively Metropolised Monte Carlo light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), November 2019. doi: 10.1145/3355089.3356578.

[13] Benedikt Bitterli and Wojciech Jarosz. Selectively Metropolised Monte Carlo light transport simulation. *ACM Transactions on Graphics (TOG)*, 38(6):1–10, 2019.

[14] Benedikt Bitterli, Wenzel Jakob, Jan Novák, and Wojciech Jarosz. Reversible jump Metropolis light transport using inverse mappings. *ACM Transactions on Graphics*, 37(1):1:1–1:12, January 2018. ISSN 0730-0301. doi: 10/gd52ph.

[15] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (TOG)*, 39(4):148–1, 2020.

[16] George E.P. Box and Mervin E. Muller. A note on the generation of random normal deviates. *The annals of mathematical statistics*, 29(2):610–611, 1958.

[17] Jeremiah U. Brackbill, Douglas B. Kothe, and Hans M. Ruppel. FLIP: a low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48(1):25–38, 1988.

[18] Robert Bridson. *Fluid Simulation for Computer Graphics*. AK Peters, Ltd., Boca Raton, 2nd edition, September 2015. ISBN 978-1-4822-3283-7.

[19] Robert Bridson, Jim Houriham, and Marcus Nordenstam. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics (TOG)*, 26(3), 2007.

[20] Tyson Brochu, Todd Keeler, and Robert Bridson. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Citeseer, 2012.

[21] Stephen Brooks. Markov chain Monte Carlo method and its application. *Journal of the royal statistical society: series D (the Statistician)*, 47(1):69–100, 1998.

[22] Barbara Busnello, Franco Flandoli, and Marco Romito. A probabilistic representation for the vorticity of a three-dimensional viscous fluid and for general systems of parabolic equations. *Proceedings of The Edinburgh Mathematical Society*, 48, 2005. doi: 10.1017/S0013091503000506.

[23] Anne Campion-Renson and Marcel J Crochet. On the stream function-vorticity finite element solutions of Navier-Stokes equations. *International Journal for Numerical Methods in Engineering*, 12(12), 1978.

[24] Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph.*, 36(4), jul 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073601.

[25] Albert Chern, Felix Knöppel, Ulrich Pinkall, Peter Schröder, and Steffen Weißmann. Schrödinger's smoke. *ACM Transactions on Graphics (TOG)*, 35(4), 2016.

[26] Alexandre Joel Chorin. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 57, 1973.

[27] J. Andrés Christen and Colin Fox. Markov Chain Monte Carlo using an approximation. *Journal of Computational and Graphical Statistics*, 14(4):795–810, 2005. doi: 10.1198/106186005X76983.

[28] Ege Ciklabakkal, Adrien Gruson, Iliyan Georgiev, Derek Nowrouzezahrai, and Toshiya Hachisuka. Single-pass stratified importance resampling. *Computer Graphics Forum (Proceedings of EGSR)*, 41(4), 2022. doi: 10.1111/cgf.14585.

[29] Pascal Clausen, Martin Wicke, Jonathan R. Shewchuk, and James F. O'Brien. Simulating liquids and solid-liquid interactions with lagrangian meshes. *ACM Transactions on Graphics (TOG)*, 32(2), 2013.

[30] David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tracing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):1186–1195, July 2005. ISSN 0730-0301. doi: 10/b3xtrn.

[31] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022. URL `http://www.blender.org`.

[32] Peter Constantin and Gautam Iyer. A stochastic Lagrangian representation of the 3-dimensional incompressible Navier-Stokes equations. *Communications on Pure and Applied Mathematics*, 61, 2007.

[33] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics (TOG)*, 5(1), 1986.

[34] Georges-Henri Cottet and Petros D. Koumoutsakos. *Vortex methods: Theory and practice*, volume 8. Cambridge university press Cambridge, 2000.

[35] Benoit Couët, Oscar Buneman, and Anthony Leonard. Simulation of three-dimensional incompressible flows with a vortex-in-cell method. *Journal of Computational Physics*, 39(2):305–328, 1981. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(81)90154-6.

[36] Ana Bela Cruzeiro. Stochastic approaches to deterministic fluid dynamics: A selective review. *Water*, 12(3), 2020. doi: 10.3390/w12030864.

[37] Pierre Del Moral. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 325(6):653–658, 1997.

[38] Freddy Delbaen, Jinniao Qiu, and Shanjian Tang. Forward-backward stochastic differential systems associated to Navier-Stokes equations in the whole space. *Stochastic Processes and their Applications*, 125, 2015.

[39] Mathieu Desbrun and Marie-Paule Gascuel. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation'96*. Springer, 1996.

[40] Simon Duane, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987. doi: 10.1016/0370-2693(87)91197-X.

[41] Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics (TOG)*, 26(1), 2007.

[42] Boris Stepanovich Elepov and Guennady Alekseevich Mikhailov. Solution of the dirichlet problem for the equation δu-cu=- q by a model of "walks on spheres". *USSR Computational Mathematics and Mathematical Physics*, 9(3):194–204, 1969.

[43] Lawrence C. Evans. Partial differential equations. *Graduate studies in mathematics*, 19(4):7, 1998.

[44] Shaohua Fan. *Sequential Monte Carlo methods for physically based rendering*. The University of Wisconsin-Madison, 2006.

[45] Shaohua Fan, Stephen Chenney, Bo Hu, Kam-Wah Tsui, and Yu-chi Lai. Optimizing Control Variate Estimators for Rendering. *Computer Graphics Forum*, 2006. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2006.00954.x.

[46] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Annual Conference Series (Proceedings of SIGGRAPH)*, pages 15–22, New York, NY, USA, August 2001. ACM Press. ISBN 978-1-58113-374-5. doi: 10/dgzhb8.

[47] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

[48] Nicholas I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1995. doi: 10.1017/CBO9780511564345.

[49] George Fishman. *Monte Carlo: Concepts, algorithms, and applications*. Springer Science & Business Media, 2013.

[50] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

[51] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical models and image processing*, 58(5):471–483, 1996.

[52] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *Journal of applied statistics*, 20(5-6):25–62, 1993.

[53] Robert A. Gingold and Joseph J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[54] Mark Girolami, Ben Calderhead, and Siu A. Chin. Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods. *J. of the Royal Statistical Society, Series B (Methodological)*, 2011. doi: 10.1111/j.1467-9868.2010.00765.x.

[55] Cindy M Goral, Kenneth E Torrance, Donald P Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. *ACM SIGGRAPH computer graphics*, 18(3):213–222, 1984.

[56] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113, 1993.

[57] Peter J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, December 1995. ISSN 0006-3444. doi: 10/bt2s2t.

[58] Peter J. Green and Antonietta Mira. Delayed rejection in reversible jump Metropolis–Hastings. *Biometrika*, 88(4):1035–1053, 2001. doi: 10.1093/biomet/88.4.1035.

[59] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[60] Ulf Grenander and Michael I. Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

[61] Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. Variance-aware multiple importance sampling. *ACM Trans. Graph. (SIGGRAPH Asia 2019)*, 38(6), 2019. doi: 10.1145/3355089.3356515.

[62] Adrien Gruson, Rex West, and Toshiya Hachisuka. Stratified Markov Chain Monte Carlo Light Transport. *Computer Graphics Forum*, 2020. ISSN 1467-8659. doi: 10.1111/cgf.13935.

[63] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7:223–242, 1998. doi: 10.2307/3318737.

[64] Heikki Haario, Marko Laine, Antonietta Mira, and Eero Saksman. DRAM: Efficient adaptive MCMC. *Statistics and Computing*, 16(4):339–354, Dec 2006. ISSN 1573-1375. doi: 10.1007/s11222-006-9438-0.

[65] Toshiya Hachisuka. Combined Lagrangian-Eulerian approach for accurate advection. In *ACM SIGGRAPH 2005 Posters*, pages 114–es. ACM New York, NY, USA, 2005.

[66] Toshiya Hachisuka and Henrik Wann Jensen. Robust adaptive photon tracing using photon path visibility. *ACM Transactions on Graphics*, 30(5):114:1–114:11, October 2011. ISSN 0730-0301. doi: 10/fpwzq9.

[67] Toshiya Hachisuka, Anton S. Kaplanyan, and Carsten Dachsbacher. Multiplexed Metropolis light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4):100:1–100:10, July 2014. ISSN 0730-0301. doi: 10/f6cswv.

[68] John Michael Hammersley and Keith William Morton. A new Monte Carlo technique: Antithetic variates. *Mathematical Proceedings of the Cambridge Philosophical Society*, 52(03):449–475, July 1956. ISSN 1469-8064. doi: 10/dshxdn.

[69] Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. Frequency domain normal map filtering. *ACM Transactions on Graphics*, 26(3):28, 2007. doi: 10.1145/1276377.1276412.

[70] Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. Improved half vector space light transport. *Computer Graphics Forum*, 34(4):65–74, July 2015. ISSN 0167-7055. doi: 10/gfzv83.

[71] Francis H Harlow and J Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The physics of fluids*, 8(12): 2182–2189, 1965.

[72] Wilfred K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970. ISSN 0006-3444. doi: 10/dkbmcf.

[73] Eric Heitz. Can't invert the CDF? the triangle-cut parameterization of the region under the curve. *Computer Graphics Forum*, 2020. ISSN 1467-8659. doi: 10.1111/cgf.14058.

[74] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[75] David A.B. Hyde and Ronald Fedkiw. A unified approach to monolithic solid-fluid coupling of sub-grid and more resolved solids. *Journal of Computational Physics*, 390: 490–526, 2019.

[76] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit incompressible SPH. *IEEE transactions on visualization and computer graphics*, 20(3):426–435, 2013.

[77] Wenzel Jakob. Mitsuba renderer, 2013. URL `http://www.mitsuba-renderer.org`.

[78] Wenzel Jakob and Steve Marschner. Manifold exploration: A Markov chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 31(4):58:1–58:13, July 2012. ISSN 0730-0301. doi: 10/gfzq4p.

[79] Y. Le Jan and A. S. Sznitman. Stochastic cascades and 3-dimensional navier–stokes equations. *Probability Theory and Related Fields*, 109, 1997.

[80] Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. Radiance caching for participating media. *ACM Transactions on Graphics (TOG)*, 27(1):1–11, 2008.

[81] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015.

[82] Herman Kahn and Theodore E Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series*, 12:27–30, 1951.

[83] James T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[84] Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph.*, 34(4), jul 2015. ISSN 0730-0301. doi: 10.1145/2766977.

[85] Anton S. Kaplanyan and Carsten Dachsbacher. Path space regularization for holistic and robust light transport. *Computer Graphics Forum (Proceedings of Eurographics)*, 32(2):63–72, 2013. doi: 10/gbc3p8.

[86] Anton S. Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4):102:1–102:13, July 2014. ISSN 0730-0301. doi: 10/f6cz85.

[87] Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media, 2012.

[88] Ioannis Karatzas, Ioannis Karatzas, Steven Shreve, and Steven E. Shreve. *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media, 1991.

[89] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A simple and robust mutation strategy for the Metropolis light transport algorithm. *Computer Graphics Forum*, 21(3):531–540, September 2002. ISSN 0167-7055. doi: 10/bfrsqn.

[90] Alexander Keller. Quasi-Monte Carlo image synthesis in a nutshell. In *Monte Carlo and Quasi-Monte Carlo Methods 2012*, pages 213–249. Springer, 2013.

[91] Ivo Kondapaneni, Petr Vévoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. Optimal multiple importance sampling. *ACM Transactions on Graphics*, 38(4), 2019. doi: 10.1145/3306346.3323009.

[92] Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. A survey on SPH methods in computer graphics. *Computer Graphics Forum*, 2022. ISSN 1467-8659. doi: 10.1111/cgf.14508.

[93] Jaroslav Krivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):550–561, 2005.

[94] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of Monte Carlo methods*. John Wiley & Sons, 2013.

[95] Joel Kronander, Thomas B. Schön, and Jonas Unger. Pseudo-marginal Metropolis Light Transport. In *ACM SIGGRAPH Asia Technical Briefs*, 2015. doi: 10/f3nd42.

[96] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of the International Conference on Computational Graphics and Visualization Techniques (Compugraphics)*, volume 93, pages 145–153, Alvor, Portugal, December 1993.

[97] Yu-Chi Lai, Shao Hua Fan, Stephen Chenney, and Charcle Dyer. Photorealistic image rendering with Population Monte Carlo energy redistribution. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 287–295. Eurographics Association, 2007. ISBN 978-3-905673-52-4. doi: 10.2312/EGWR/EGSR07/287-295.

[98] Mario Lazzarini. Un'applicazione del calcolo della probabilitá alla ricerca sperimentale di un valore approssimato di π. *Periodico di Matematica*, 4:140–143, 1901.

[99] Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. Gradient-domain Metropolis light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 32(4):95:1–95:12, July 2013. ISSN 0730-0301. doi: 10/gbdghd.

[100] Christiane Lemieux. Control variates. *Wiley StatsRef: Statistics Reference Online*, pages 1–8, 2014.

[101] Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. Anisotropic Gaussian mutations for Metropolis light transport through Hessian-Hamiltonian dynamics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 34(6):209:1–209:13, October 2015. ISSN 0730-0301. doi: 10/f7wrcs.

[102] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 37(6), December 2018. doi: 10.1145/3272127.3275109.

[103] Wei Li, Kai Bai, and Xiaopei Liu. Continuous-scale kinetic fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2694–2709, 2018.

[104] Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. Fast and scalable turbulent flow simulation with two-way coupling. *ACM Trans. Graph.*, 39 (4), jul 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392400.

[105] Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. Generalized resampled importance sampling: foundations of ReSTIR. *ACM Transactions on Graphics (TOG)*, 41(4):1–23, 2022.

[106] Jun S Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.

[107] Jun S. Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, March 2000. ISSN 0162-1459. doi: 10/gfkjqj.

[108] Samuel Livingstone and Mark Girolami. Information-geometric markov chain Monte Carlo methods using diffusions. *Entropy*, 16(6):3074–3102, 2014. doi: 10.3390/e16063074.

[109] Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. Langevin Monte Carlo rendering with gradient-based adaptation. *ACM Trans. Graph.*, 39(4):140, 2020.

[110] Thomas Lundgren and Petros Koumoutsakos. On the generation of vorticity at a free surface. *Journal of Fluid Mechanics*, 382:351–366, 1999.

[111] Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. Fast and versatile fluid-solid coupling for turbulent flow simulation. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021.

[112] Miles Macklin and Matthias Müller. Position based fluids. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.

[113] Sylvain Maire and Etienne Tanre. Monte Carlo approximations of the Neumann problem. *Walter de Gruyter GmbH*, 19, 2013.

[114] Zander Majercik, Thomas Mueller, Alexander Keller, Derek Nowrouzezahrai, and Morgan McGuire. Dynamic diffuse global illumination resampling. In *ACM SIGGRAPH 2021 Talks*, SIGGRAPH '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383738. doi: 10.1145/3450623.3464635.

[115] Enzo Marinari and Giorgio Parisi. Simulated tempering: a new Monte Carlo scheme. *Europhysics letters*, 19(6):451, 1992.

[116] Gisiro Maruyama. Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo*, 4(1):48–90, 1955.

[117] Tom McCombes, Cameron Johnstone, and Andrew Grant. Unsteady 3D wake modelling for marine current turbines. In *Proceedings of the 8th European Wave and Tidal Energy Conference*, 2009.

[118] Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[119] Olivier Mercier, Xi-Yuan Yin, and Jean-Christophe Nave. The characteristic mapping method for the linear advection of arbitrary sets. *SIAM Journal on Scientific Computing*, 42(3):A1663–A1685, 2020.

[120] Nicholas Metropolis. The beginning of the Monte Carlo method. *Los Alamos Science*, (15):125–130, 1987.

[121] Nicholas Metropolis and Stanisław M. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, September 1949. ISSN 01621459. doi: 10/dvn2n8.

[122] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, June 1953. ISSN 0021-9606. doi: 10/ds736f.

[123] Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. Boundary value caching for walk on spheres. *ACM Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301. doi: 10.1145/3592400.

[124] Antonietta Mira. On Metropolis–Hastings algorithms with delayed rejection. *Metron*, 59(3-4):231–241, 2001.

[125] Zackary Misso, Benedikt Bitterli, Iliyan Georgiev, and Wojciech Jarosz. Unbiased and consistent rendering using biased estimators. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022.

[126] Marek Krzysztof Misztal, Kenny Erleben, Adam Bargteil, Jens Fursund, Brian Bunch Christensen, Jakob Andreas Bærentzen, and Robert Bridson. Multiphase flow of immiscible fluids on unstructured moving meshes. *IEEE transactions on visualization and computer graphics*, 20(1), 2013.

[127] Joe J. Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30(1):543–574, 1992.

[128] Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-preserving integrators for fluid animation. *ACM Trans. Graph*, 28, 2009. doi: 10.1145/1531326.1531344.

[129] Matthias Müller, David Charypar, and Markus H Gross. Particle-based fluid simulation for interactive applications. In *Symposium on Computer animation*, 2003.

[130] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2): 109–118, 2007.

[131] Mervin E. Muller. Some continuous Monte Carlo methods for the Dirichlet problem. *Annals of Mathematical Statistics*, 27(3), 1956.

[132] Thomas Müller, Markus Gross, and Jan Novák. Practical path guiding for efficient light-transport simulation. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 36(4):91–100, June 2017. doi: 10/gbnvrs.

[133] Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics (ToG)*, 38(5): 1–19, 2019.

[134] Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. Neural control variates. *ACM Transactions on Graphics (TOG)*, 39(6):1–19, 2020.

[135] Iain Murray, Ryan Adams, and David MacKay. Elliptical slice sampling. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 541–548, May 2010.

[136] Mohammad Sina Nabizadeh, Albert Chern, and Ravi Ramamoorthi. Kelvin transformations for simulations on infinite domains. *ACM Transactions on Graphics (TOG)*, 40(4), 2021.

[137] Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. Covector fluids. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022.

[138] C.L.M.H. Navier. Mémoire sur les lois du mouvement des fluides. *Mémoires de l'Académie Royale des Sciences de l'Institut de France*, 6(1823):389–440, 1823.

[139] Radford M. Neal. MCMC using Hamiltonian dynamics. In Steve Brooks, Andrew Gelman, Galin L. Jones, and Xiao-Li Meng, editors, *Handbook of Markov Chain Monte Carlo*, volume 2, pages 113–162. Chapman & Hall/CRC, May 2011. ISBN 1-4200-7941-7. doi: 10.1201/b10905-6.

[140] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), November 2019. doi: 10.1145/3355089.3356498.

[141] Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum (Proceedings of Eurographics State of the Art Reports)*, 37(2):551–576, May 2018. ISSN 0167-7055. doi: 10/gd2jqq.

[142] Bernt Oksendal. *Stochastic differential equations: An introduction with applications*. Springer Science & Business Media, 2013.

[143] Hisanari Otsu, Yonghao Yue, Qiming Hou, Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. Replica exchange light transport on relaxed distributions. In *ACM SIGGRAPH 2013 Posters*, SIGGRAPH '13, pages 106:1–106:1. ACM, 2013. ISBN 978-1-4503-2342-0. doi: 10.1145/2503385.2503501.

[144] Hisanari Otsu, Anton S. Kaplanyan, Johannes Hanika, Carsten Dachsbacher, and Toshiya Hachisuka. Fusing state spaces for Markov chain Monte Carlo rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 36(4):74:1–74:10, July 2017. ISSN 0730-0301. doi: 10/gbxjs9.

[145] Hisanari Otsu, Johannes Hanika, Toshiya Hachisuka, and Carsten Dachsbacher. Geometry-aware Metropolis light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 37(6):278:1–278:11, 2018. ISSN 0730-0301. doi: 10/gf2r3t.

[146] Y. Ouyang, S. Liu, M. Kettunen, M. Pharr, and J. Pantaleoni. ReSTIR GI: Path resampling for real-time path tracing. *Computer Graphics Forum*, 40(8):17–29, 2021. doi: https://doi.org/10.1111/cgf.14378.

[147] Jacopo Pantaleoni. Charted Metropolis light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 36(4):75:1–75:14, July 2017. ISSN 0730-0301. doi: 10/gfzq78.

[148] Sang Il Park and Myoung Jun Kim. Vortex fluid for gaseous phenomena. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2005.

[149] M.F. Peeters, W.G. Habashi, and E.G. Dueck. Finite element stream function-vorticity solutions of the incompressible Navier-Stokes equations. *International journal for numerical methods in fluids*, 7(1), 1987.

[150] Tobias Pfaff, Nils Thuerey, and Markus Gross. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics (TOG)*, 31(4), 2012.

[151] Matt Pharr. Guest editor's introduction: Special issue on production rendering. *ACM Transactions on Graphics (TOG)*, 28(3), 2018. doi: 10.1145/3212511.

[152] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, Cambridge, MA, 3rd edition, 2016. ISBN 978-0-12-800645-0.

[153] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2018.

[154] Yang Qi, Dario Seyb, Benedikt Bitterli, and Wojciech Jarosz. A bidirectional formulation for Walk on Spheres. *Computer Graphics Forum (Proceedings of EGSR)*, 41(4), July 2022. ISSN 1467-8659. doi: 10.1111/cgf.14586.

[155] Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. Efficient and conservative fluids using bidirectional mapping. *ACM Transactions on Graphics (TOG)*, 38(4), 2019.

[156] Amir Hossein Rabbani, Jean-Philippe Guertin, Damien Rioux-Lavoie, Arnaud Schoentgen, Kaitai Tong, Alexandre Sirois-Vigneux, and Derek Nowrouzezahrai. Compact Poisson filters for fast fluid simulation. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[157] Damien Rioux-Lavoie, Joey Litalien, Adrien Gruson, Toshiya Hachisuka, and Derek Nowrouzezahrai. Delayed rejection Metropolis light transport. *ACM Transactions on Graphics (TOG)*, 39(3):1–14, 2020.

[158] Damien Rioux-Lavoie, Ryusuke Sugimoto, Tümay Özdemir, Naoharu H Shimada, Christopher Batty, Derek Nowrouzezahrai, and Toshiya Hachisuka. A Monte Carlo method for fluid simulation. *ACM Transactions on Graphics (TOG)*, 41(6):1–16, 2022.

[159] Christian Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, 2005. doi: 10.1007/978-1-4757-4145-2.

[160] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 1999.

[161] Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009. doi: 10.1198/jcgs.2009.06134.

[162] Jeff Rosenthal. *A First Look at Rigorous Probability Theory*. World Scientific Publishing Company, 2010.

[163] Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. Image-space control variates for rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 35(6): 169:1–169:12, November 2016. ISSN 0730-0301. doi: 10/f9cphw.

[164] Takahiro Sato, Christopher Batty, Takeo Igarashi, and Ryoichi Ando. Spatially adaptive long-term semi-Lagrangian method for accurate velocity advection. *Computational Visual Media*, 4(3), 2018.

[165] Rohan Sawhney and Keenan Crane. Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains. *ACM Transactions on Graphics (TOG)*, 39(4), 2020.

[166] Rohan Sawhney, Daqi Lin, Markus Kettunen, Benedikt Bitterli, Ravi Ramamoorthi, Chris Wyman, and Matt Pharr. Decorrelating ReSTIR samplers via MCMC mutations. *ACM Transactions on Graphics*, 2022.

[167] Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. Grid-free Monte Carlo for PDEs with spatially varying coefficients. *ACM Transactions on Graphics (TOG)*, 41(4), 2022.

[168] Rohan Sawhney, Bailey Miller, Ioannis Gkioulekas, and Keenan Crane. Walk on stars: A grid-free Monte Carlo method for PDEs with Neumann boundary conditions. *ACM Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301. doi: 10.1145/3592398.

[169] Arnaud Schoentgen. *Tools for Fluid Simulation Control in Computer Graphics*. PhD thesis, Poitiers, 2021.

[170] Benjamin Segovia, Jean-Claude Iehl, and Bernard Péroche. Coherent Metropolis light transport with multiple-try mutations. Technical Report RR-LIRIS-2007-015, Universite Lyon, Lyon, France, April 2007.

[171] Dino Sejdinovic, Heiko Strathmann, Maria Lomeli Garcia, Christophe Andrieu, and Arthur Gretton. Kernel adaptive Metropolis-Hastings. In *International conference on machine learning*, pages 1665–1673. PMLR, 2014.

[172] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. In *ACM SIGGRAPH*, 2005.

[173] Chris Sherlock, Andrew Golightly, and Daniel A. Henderson. Adaptive, delayed-acceptance MCMC for targets with expensive likelihoods. *Journal of Computational and Graphical Statistics*, 26(2):434–444, 2017. doi: 10.1080/10618600.2016.1231064.

[174] N. H. Shimada and T. Hachisuka. Quantum coin method for numerical integration. *Computer Graphics Forum*, 2020. ISSN 1467-8659. doi: 10.1111/cgf.14015.

[175] Martin Šik and Jaroslav Křivánek. Survey of Markov Chain Monte Carlo methods in light transport simulation. *IEEE Transactions on Visualization and Computer Graphics*, 2018. ISSN 1077-2626. doi: 10.1109/TVCG.2018.2880455.

[176] Nikolai A Simonov. Walk-on-Spheres algorithm for solving third boundary value problem. *Applied Mathematics Letters*, 64:156–161, 2017.

[177] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible SPH. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605587264. doi: 10.1145/1576246.1531346.

[178] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999.

[179] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 129–136, 1995.

[180] George Gabriel Stokes. *On the Theories of the Internal Friction of Fluids in Motion, and of the Equilibrium and Motion of Elastic Solids*, volume 1 of *Cambridge Library Collection - Mathematics*, page 75–129. Cambridge University Press, 2009. doi: 10.1017/CBO9780511702242.005.

[181] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.

[182] Robert H. Swendsen and Jian-Sheng Wang. Replica Monte Carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

[183] László Szirmay-Kalos and László Szécsi. Improved stratification for Metropolis light transport. *Computers & Graphics*, 68:11–20, 2017. doi: 10.1016/j.cag.2017.07.032.

[184] Justin F. Talbot, David Cline, and Parris Egbert. Importance Resampling for Global Illumination. In *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*, pages 139–146. Eurographics Association, June 2005. ISBN 3-905673-23-1. doi: 10/gfzsm2.

[185] Andre Pradhana Tampubolon, Theodore Gast, Gergely Klár, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

[186] Geoffrey Ingram Taylor and Albert Edward Green. Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences*, 158(895):499–521, 1937.

[187] Jerry Tessendorf and Brandon Pelfrey. The characteristic map for fast and efficient VFX fluid simulations. In *Computer Graphics International Workshop on VFX, Computer Animation, and Stereo Movies. Ottawa, Canada*, 2011.

[188] Luke Tierney and Antonietta Mira. Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine*, 18(17-18):2507–2515, 1999. doi: 10.1002/(sici)1097-0258(19990915/30)18:17/18⟨2507::aid-sim272⟩3.0.co;2-j.

[189] Miquel Trias, Alberto Vecchio, and John Veitch. Delayed rejection schemes for efficient Markov-Chain Monte-Carlo sampling of multimodal distributions. *arXiv preprint arXiv:0904.2207*, 2009.

[190] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Thesis, Stanford University, December 1997.

[191] Eric Veach and Leonidas J. Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*, pages 145–167. Springer-Verlag, 1995. ISBN 978-3-642-87825-1. doi: 10/gfznbh.

[192] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Annual Conference Series (Proceedings of SIGGRAPH)*, volume 29, pages 419–428. ACM Press, August 1995. ISBN 978-0-89791-701-8. doi: 10/d7b6n4.

[193] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Annual Conference Series (Proceedings of SIGGRAPH)*, volume 31, pages 65–76. ACM Press, August 1997. ISBN 978-0-89791-896-1. doi: 10/bkjqj4.

[194] Eric Veach and Leonidas J Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76, 1997.

[195] John Von Neumann. 13. various techniques used in connection with random digits. *Appl. Math Ser*, 12(36-38):3, 1951.

[196] Jiří Vorba and Jaroslav Křivánek. Adjoint-driven Russian roulette and splitting in light transport simulation. *ACM Transactions on Graphics (TOG)*, 35(4), 2016. doi: 10.1145/2897824.2925912.

[197] Martin Šik and Jaroslav Křivánek. Improving global exploration of MCMC light transport simulation. In *ACM SIGGRAPH 2016 Posters*, SIGGRAPH '16, pages 50:1–50:2, 2016. doi: 10.1145/2945078.2945128.

[198] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 85–92, 1988.

[199] Rex West, Iliyan Georgiev, Adrien Gruson, and Toshiya Hachisuka. Continuous multiple importance sampling. *ACM Transactions on Graphics (TOG)*, 39(4):136–1, 2020.

[200] Rex West, Iliyan Georgiev, and Toshiya Hachisuka. Marginal multiple importance sampling. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–8, 2022.

[201] Shiying Xiong, Rui Tao, Yaorui Zhang, Fan Feng, and Bo Zhu. Incompressible flow simulation of vortex segment clouds. *ACM Transactions on Graphics (TOG)*, 40(4), 2021.

[202] Dongbin Xiu and George Em Karniadakis. A semi-Lagrangian high-order method for Navier–Stokes equations. *Journal of computational physics*, 172(2):658–684, 2001.

[203] Shuqi Yang, Shiying Xiong, Yaorui Zhang, Fan Feng, Jinyuan Liu, and Bo Zhu. Clebsch gauge fluid. *ACM Transactions on Graphics (TOG)*, 40(4), 2021.

[204] Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. Specular manifold sampling for rendering high-frequency caustics and glints. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 39(4), July 2020. doi: 10.1145/3386569.3392408.

[205] Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. Monte Carlo estimators for differential light transport. *ACM Transactions on Graphics (TOG)*, 40 (4):1–16, 2021.

[206] Cheng Zhang, Zhao Dong, Michael Doggett, and Shuang Zhao. Antithetic sampling for Monte Carlo differentiable rendering. *ACM Transactions on Graphics (TOG)*, 40(4):1–12, 2021.

[207] Xinxin Zhang and Robert Bridson. A PPPM fast summation method for fluids and beyond. *ACM Transactions on Graphics (TOG)*, 33(6), 2014.

[208] Yichuan Zhang and Charles Sutton. Quasi-newton methods for Markov chain Monte Carlo. *Advances in Neural Information Processing Systems*, 24, 2011.

[209] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)*, 24(3), 2005.

[210] Károly Zsolnai and László Szirmay-Kalos. Automatic parameter control for Metropolis light transport. In M.-A. Otaduy and O. Sorkine, editors, *Proceedings of Eurographics Short Papers*. The Eurographics Association, 2013. doi: 10/gf2r3s.