Monte Carlo Geometry Processing: A Grid-Free Approach to Solving Partial Differential Equations on Volumetric Domains

Rohan Sawhney

CMU-CS-24-125
March, 2024

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Keenan Crane (Chair)
Ioannis Gkioulekas
Matthew O'Toole
Gautam Iyer (CMU Math)
Matt Pharr (NVIDIA)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by National Science Foundation awards IIS1943123 and CCF1717320, the David and Lucille Packard Foundation award 201868047, Apple, Inc. award 1012669, OCULUS VR LLC award PO1075912, gifts from nTop and Disney, and an NVIDIA Graduate Fellowship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Abstract

This thesis develops Monte Carlo algorithms based on the *walk on spheres (WoS)* method to reliably solve fundamental partial differential equations (PDEs) like the Poisson equation on geometrically complex domains. Elliptic PDEs are a basic building block of algorithms and applications throughout science, engineering, and geometric computing. Yet despite decades of research on methods for solving such PDEs, conventional solvers still struggle to deal with the level of geometric complexity found in the natural world. A constant challenge is the need for spatial discretization, which traditionally involves dividing the domain into a high-quality volumetric mesh or grid to perform PDE-based analysis. Unfortunately, this approach does not scale well to modern computer architectures as it is inherently sequential and memory intensive. It also falters when dealing with imperfect data containing poorly-shaped elements or self-intersections. These shortcomings together hinder the ability of scientists, engineers and designers to analyze geometric data and iterate on designs.

Walk on spheres makes a radical departure from conventional PDE solvers by reformulating the problem in terms of recursive integral equations that can be solved using the Monte Carlo method, allowing it to avoid volumetric mesh generation and function space approximation altogether. Furthermore, since these integral equations closely resemble those found in light transport theory, one can leverage deep knowledge from Monte Carlo rendering to build new algorithms for solving PDEs.

In this work, we take inspiration from rendering to generalize WoS to solve a much broader set of linear elliptic PDEs on solid regions of $\mathbb{R}^N$. We develop complete "black box" solvers encompassing integration, variance reduction and acceleration. Our solvers share many benefits with Monte Carlo methods from rendering: no volumetric meshing, trivial parallelism, output-sensitive evaluation of the PDE solution and its gradient without the need to solve a globally-coupled system of equations, and the ability to handle geometric data of size and complexity that is essentially hopeless for grid-based techniques.

Keywords: partial differential equations, Monte Carlo methods, geometry processing, physics-based rendering and simulation, computer graphics

Acknowledgements

Doing a PhD has been similar in many ways to taking a random walk through a maze: it took the better part of two years to stumble upon a topic that was intriguing yet underexplored. From there, it has taken another four years to unravel some of its mysteries, with a lot more yet to be uncovered! I am eternally grateful to my mentors, collaborators, family and friends who have taken this long but rewarding journey with me.

First and foremost, I would like to thank my PhD advisor, Keenan Crane, for his boldness which inspired me, his unrelenting belief which helped me stay the course, and his patience which supported me when I needed it most. Keenan has taught me so much about being driven by curiosity, not being afraid to take "big bets," and communicating ideas effectively. These learnings will stay with me for life. I would also like to express my deepest gratitude to Bailey Miller, Ioannis Gkioulekas, Dario Seyb and Wojciech Jarosz for being absolutely amazing collaborators and friends. The enthusiasm you have shared, the time you have invested, and the hardwork you have put into this research has opened new doors that I, at times, did not think existed. I feel extremely lucky to have been a part of teams that have functioned so well together, and from whom I have learned so much. I truly enjoyed our collaborations, and I look forward to many more in the future!

I would like to thank Matt Pharr, Rasmus Tamstorf, Michael Mascagni, Gautam Iyer, Matthew O'Toole and Bradley Rothenberg for their wholehearted support over the course of this journey. Your insights and encouragement have gone above and beyond anything I could have asked for.

The Geometry Collective, and Carnegie Mellon more generally, provided a fantastic environment to work and build friendships in. It was a joy to have spent time with Nick Sharp, Chris Yu, Kai Ye, Mark Gillespie, Etienne Corman, Nicole Feng, Olga Guțan, Hossein Baktash, Yousuf Soliman, Mina Konakovic, Nimo Wode Ni, Josua Sassen, Max Slater, Daniel Li, Ethan Lu, Vidya Narayanan, Ravi Mullapudi, Evan Shimizu, Karima Ma, Arjun Teh, Jim McCann, Kayvon Fatahalian, Angela Jiang, John Wright, Nika Haghtalab, Naama Ben-David, David Wajc, Colin White, Noam Brown, Nic Resch, Tanya Marwah, and Gaurav Pathak. I would like to give a special shoutout to Christina Contreras and Deb Cavlovich, who have personally ensured over the years that everything runs smoothly for all students at CMU, myself included.

During my PhD, I was fortunate to have interned with three top-notch teams who helped greatly broaden my perspective: thank you Matt Pharr, Chris Wyman, Daqi Lin, Markus Kettunen, Benedikt Bitterli, Ravi Ramamoorthi, Marco Salvi, Eugene d'Eon, Lifan Wu, Aaron Lefhon, and Ken Museth for welcoming me at NVIDIA, and teaching me so much more about rendering and physics. Thank you Noam Aigerman, Danny Kaufman, Vova Kim, and Nathan Carr for deepening my understanding of geometry processing at Adobe, and thank you Bradley Rothenberg, Blake Courter, Suraj Musuvathy, Johann Korndoerfer, and Trevor Laughlin for giving me the opportunity to experience the exciting and fast-paced world of startups at nTop!

Prior to starting my PhD, I have my undergraduate and high school advisors, Eitan Grinspun, Michael Reed and Vipul Mehra, to thank for captivating me with their infectious excitement and enthusiasm for physics and computer graphics. I would also like to thank my friends,

Kunal Mehta, Zan Gilani, Hong-Yi TuYe, Bitania Wondimu, Mihika Barua, Vishal Alluri, Alex Xu, Kareem Carryl, Henrique Maia, Udit Toshniwal, Ishaan Jalan, Veer Bhartiya, Bhavik Merchant, Ankit Vaish, Karan Kaushik, and Akshay Shah, for enduring and encouraging these passions of mine, as well as my colleagues at IrisVR, Shane Scranton, Nate Beatty, Jack Donovon, Robin Kim, George Valdes, Ailyn Mendoza, Ana Garcia, Tyler Holf, Ezra Miller and Matt Gooding, for giving me the opportunity to gain my first real-world work experience.

Last, and most importantly, the *biggest* thank you to my wife and lifelong friend Ellen Vitercik, my loving parents Chhaya and Rajesh Sawhney, my steadfast brother Arjun Sawhney, and my adorable cat Laddu for always being by my side! Through the ups and the downs, you have provided me with a platform like no other to pursue my goals. Words can hardly do justice to the gratitude I feel for having such a supportive family.

A Note About Content

The text and figures in this thesis have significant overlap with the papers published by my collaborators and I over the course of my PhD. The following publications, used with permission, are the original source of much of the content here:

1. Rohan Sawhney and Keenan Crane. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Transactions on Graphics* (2020).

2. Rohan Sawhney*, Dario Seyb*, Wojciech Jarosz†, and Keenan Crane†. Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients. *ACM Transactions on Graphics* (2022).

3. 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 Transactions on Graphics* (2023).

4. Bailey Miller*, Rohan Sawhney*, Keenan Crane†, and Ioannis Gkioulekas†. Boundary Value Caching for Walk on Spheres. *ACM Transactions on Graphics* (2023).

5. Bailey Miller*, Rohan Sawhney*, Keenan Crane†, and Ioannis Gkioulekas†. Walkin' Robin: Walk on Stars with Robin Boundary Conditions. *ACM Transactions on Graphics* (2024).

The symbols * and † denote equal contribution.

Contents

Chapter 1

Introduction

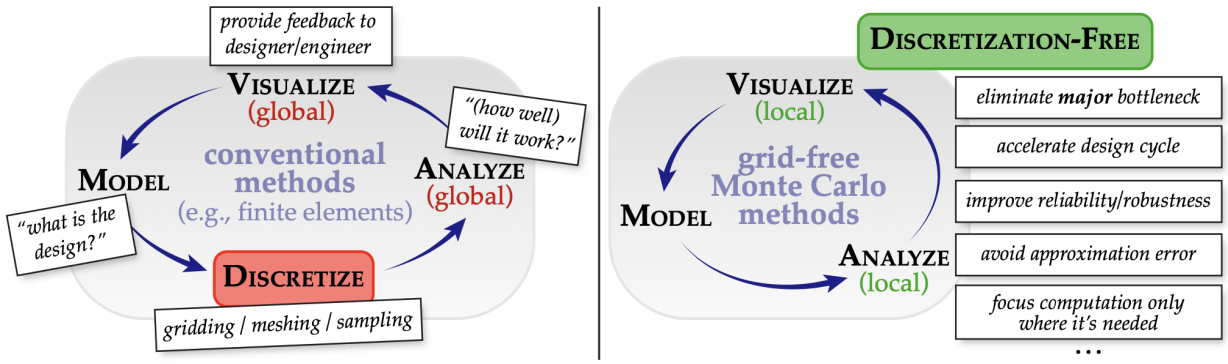


Figure 1.1: Grid-free Monte Carlo methods developed in this work can significantly speedup the engineering design cycle by eliminating the major bottleneck of discretization in conventional solvers for partial differential equations (left). Lifting the dependence on discretization and global solves improves robustness, and like Monte Carlo ray tracing allows computation effort to be focused entirely on local regions of interest (right).

1.1 Motivation

The ability to accurately model and analyze large amounts of geometric information is fundamental to many scientific and engineering disciplines, ranging from geology to medicine, and autonomous driving to industrial design. Subtle differences in fine-scale geometry can have a major impact on the large-scale behavior of many physical systems—consider, for instance, the influence of millions of tiny alveolar air sacs on the dispersion of oxygen in the lungs, the impact of intricate grille patterns on the acoustic performance of a microphone, and the role of all the wiring and plumbing in a building information model on the thermal response of a structure. At all scales, detailed and irregular geometry plays a crucial role in our ability to understand the function of a physical system, evaluate its performance and predict its failure modes.

Techniques based on *partial differential equations (PDEs)* provide powerful tools for analyzing many such physical phenomena. Unfortunately, despite the drastic increase in our ability to capture and generate complex geometric models in recent years, conventional methods for solving PDEs are not yet at a stage where they "just work" on problems of real-world complexity. Basic tasks involving PDEs still entail careful preprocessing or parameter tuning, and solvers regularly exhibit poor scaling in time or memory. Even more broadly, models of real physical systems must often integrate disparate phenomena, such as light transport and heat transfer, which classically demand very different computational tools that do not "play well together". For these reasons, there remains a large divide between our ability to *visualize* and *simulate* the natural world, and we tend to shy away from simulating it at its original level of complexity by either making gross approximations via model reduction and homogenization—or by tempering our ambition.

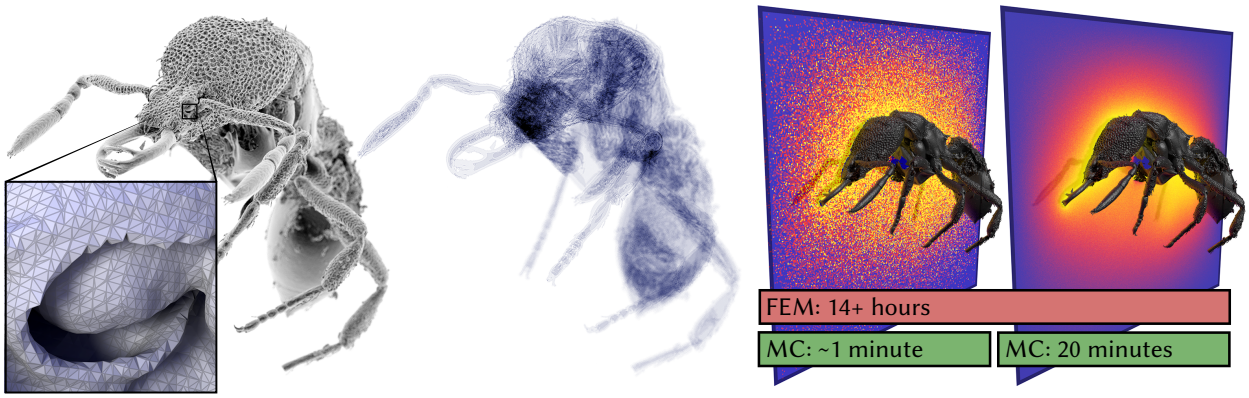


Figure 1.2: Real-world geometry has rich surface detail (left) and intricate internal structure (center). On such domains, FEM-based geometric algorithms struggle to mesh, setup, and solve PDEs—in this case taking more than 14 hours and 30GB of memory just for a basic Poisson equation. Our Monte Carlo solver uses about 1GB of memory and takes less than a minute to provide a preview (center right) that can then be progressively refined (far right).

A significant issue with traditional numerical methods for solving PDEs, such as the *finite element method (FEM)*, is the end-to-end cost of the pipeline: even if the FEM solver is fast, one must first convert the boundary description of the input geometry into a "simulation-ready" volumetric mesh. Unfortunately, meshing is brittle and often requires intervention from expert engineers: it can easily fail on data with minor imperfections such as self-intersections, and a few badly-shaped elements can spoil an entire FEM solution. Furthermore, state-of-the-art robust meshing algorithms can be extremely time consuming and memory intensive on intricate geometric domains (*e.g.*, Fig. 1.2, 8.1 & 8.4). Though performance can be improved with more powerful processors, the inherently sequential nature of mesh generation makes it difficult to truly leverage increasingly parallel architectures. Moreover, even when meshing succeeds, important geometric detail can be lost in the discretization (Fig. 1.3 & 8.1), which results in aliasing artifacts in the PDE solution (Fig. 8.9 & 8.15).

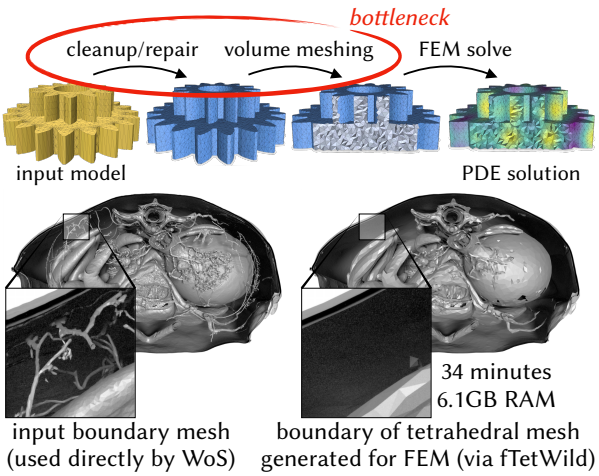


Figure 1.3: The bottleneck in conventional PDE solvers like FEM is often not the solve itself, but rather the cost of meshing (top). Robust meshing algorithms [105] can also sacrifice spatial detail–here destroying key features like blood vessels (bottom).

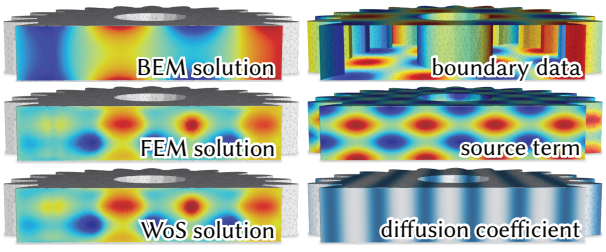


*Figure 1.4: Unlike FEM and Monte Carlo, traditional BEM does not consider volumetric functions (*e.g.*, coefficients and source terms) that affect the PDE solution.*

The cost and difficulty of mesh generation for conventional FEM sparked the development of *meshless FEM* and *boundary element methods (BEM)*, though all of these approaches suffer from a common problem: the need to spatially discretize (*e.g.*, mesh or point-sample) the domain interior. BEM must be integrated with volumetric methods like FEM to han-

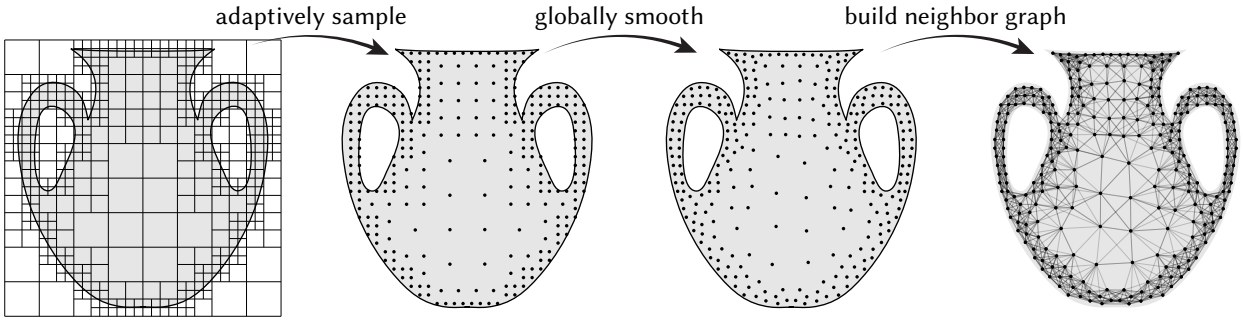


Figure 1.5: So-called "meshless" methods still perform a process akin to global meshing, which can result in spatial aliasing of fine features. One ends up with a mesh-like structure that must satisfy stringent sampling criteria to avoid numerical blowup, and must solve a large globally coupled linear system. Figure adapted from [199, Figure 6].

dle interior terms (Fig. 1.4), while even so-called meshless methods must carefully place nodes over the *entire* domain (Fig. 1.5). Furthermore, for problems with volumetric heterogeneities, the discretization must be carefully adapted to regions where material coefficients exhibit fine detail (Fig. 5.1 & 8.10). As a result, many engineering and scientific disciplines today suffer from non-interactive workflows due to the bottleneck of needing to first convert large amounts of geometric information into a form suitable for PDE-based analysis and simulation (Fig. 1.1, *left*). This limits experts and non-experts alike from making sense of their data, and analyzing it quickly to solve geometric problems.

1.2 Approach & Scope

To address these challenges, this thesis makes a major break from conventional PDE solvers, and instead explores how to solve basic, yet fundamental PDEs in geometric computing with *grid-free Monte Carlo methods* based on Muller's *walk on spheres (WoS)* algorithm [174]. This shift mirrors an analogous development in the 1990s for photorealistic rendering: for reasons nicely summarized by Jensen et al. [116], algorithms built around *finite element radiosity* [85] gave way to Monte Carlo ray tracing of the light transport equation [121]. A key motivation was to simulate more complex illumination, but the shift also made it possible to work with scenes of extreme geometric complexity—modern renderers today can handle trillions of effective polygons [76] and, in stark contrast to FEM, provide high-quality results without any preconditions on the input geometry. As all geometric computation boils down to simple ray intersection queries, Monte Carlo renderers also work with representations other than polygon meshes. They exhibit excellent scaling, offer a trivial parallel implementation, and allow for view-dependent evaluation. Collectively, these features of Monte Carlo rendering have helped it revolutionize industries such as film, architecture and industrial design by enabling practitioners to iterate on their designs quickly.

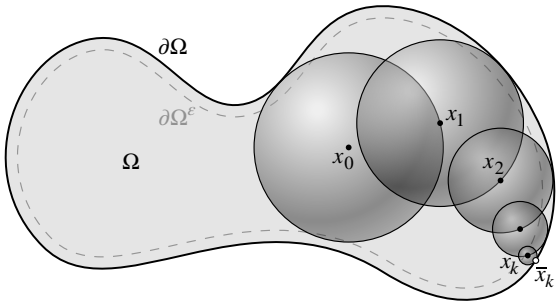


Figure 1.6: The basic idea behind walk on spheres is that at any point x, the value of a harmonic function $u(x)$ equals the average over a sphere around x [9]. Hence we recursively take a single Monte Carlo sample to estimate this average until we hit the boundary. No spatial discretization is needed since the largest empty sphere can be determined using a closest point query.

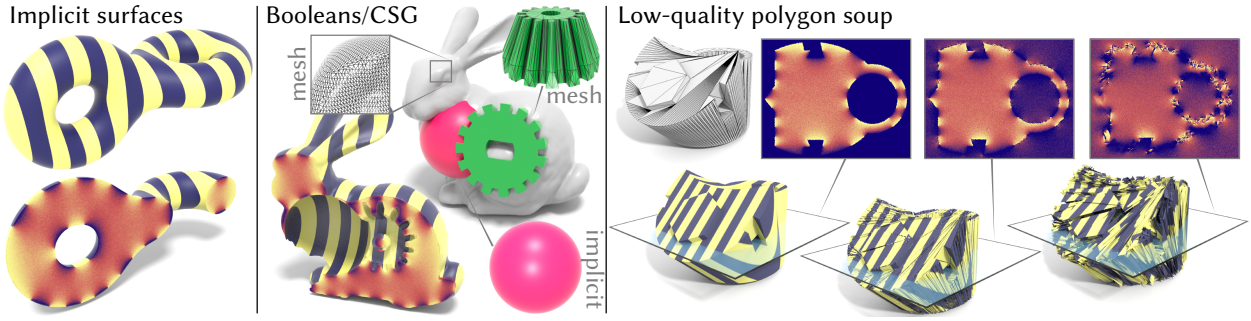


Figure 1.7: As with Monte Carlo ray tracing, grid-free Monte Carlo methods like walk on spheres can solve PDEs directly on a variety of geometric representations without needing to discretize the boundary or volume. Unlike traditional PDE solvers, WoS also exactly captures discontinuous boundary conditions and provides a meaningful solution even on geometry with poor element quality; the solution degrades gracefully in the presence of noise.

The Monte Carlo methods described in this work provide an analogous approach for geometric computing: the main idea is to express elliptic PDEs, arising in countless physical problems such as the diffusion of heat, electrostatic potentials, and incompressible fluid flows, as recursive integral equations that look a lot like the *rendering equation* [121]. This reformulation enables use of Monte Carlo integration to solve these integral equations. Samples are generated by replacing recursive ray tracing with the recursive walk on spheres algorithm (Fig. 1.6) and its generalizations, which use the uniform distribution over a sphere to *exactly* model large steps of continuous random processes such as *Brownian motion* [191]. This approach shares a number of benefits with Monte Carlo rendering:

- **Geometric Flexibility.** It can work directly with implicit surfaces, NURBS/subdivision surfaces, constructive solid geometry, procedural/instanced geometry, etc., without explicit tessellation (Fig. 1.7).
- **Geometric Robustness.** Geometry need not be watertight, manifold, nor free of self-intersections; sharp edges, small details, and thin features are exactly preserved.
- **Scalability.** The main cost is a *bounding volume hierarchy (BVH)* for distance queries, which is $O(n \log n)$ in time and memory with respect to the size of the boundary (Fig. 1.8).
- **Parallelism.** It is trivial to achieve near-perfect parallel scaling, and many operations are easily vectorized.
- **Correctness.** Since there is no discretization of space or approximation of function spaces, one obtains the exact solution in expectation, *i.e.*, error is almost entirely due to the variance of the Monte Carlo estimator, and can be reduced by simply taking more samples.
- **Adaptivity.** Adaptive sampling akin to *radiance caching* [266] can significantly reduce cost in smooth regions (Fig. 7.7); progressive sampling enables rapid previews of PDE solutions (Fig. 1.2 & 4.2).
- **Output Sensitivity.** The solution can be evaluated in local regions of interest, like a small window (Fig. 1.9 & 6.1) or a slice (Fig. 8.6), without having to first perform a global solve.
- **Compatibility.** Monte Carlo methods fit easily into standard pipelines for geometric computing, as they can be used as "black box" solvers that return reliable and accurate solution values at any given query point (*e.g.*, at mesh vertices).

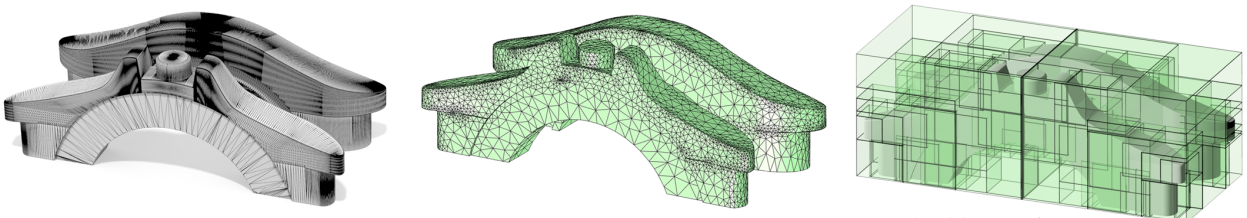


Figure 1.8: Finite element methods exhibit unpredictable performance, as models with simple geometry but poor element quality (left) can confound even robust meshing algorithms (center, via Hu et al. [105]). The Monte Carlo approach only needs to build a standard bounding volume hierarchy (right), which dramatically reduces precomputation time for solving PDEs.

Monte Carlo methods are not, however, a silver bullet. On simple domains with smooth boundary conditions, solvers like FEM are quite mature and hard to beat in terms of solve time; here Monte Carlo can be slow to eliminate high-frequency noise (though see Ch. 7 for variance reduction strategies). Yet for more complex problems, end-to-end performance depends on many factors beyond just the rate of convergence of the core solve, such as mesh generation, parallel scaling, and visualization; Ch. 8 provides an in-depth discussion on tradeoffs with conventional PDE solvers. It may also not be straightforward to formulate a Monte Carlo estimator for any given PDE. Here we do not strive for full feature compatibility with traditional solvers and all the PDEs they can solve—we instead focus on a specific class of problems, namely 2*nd order linear elliptic equations* (reviewed in Ch. 2), which power a large array of applications in geometric computing such as surface reconstruction [129, 130] and shape optimization [204] (using measurements of physical quantities like temperature [277]). Monte Carlo solvers are not limited just to WoS [101, 134, 172, 217]—we compare against some of these techniques in Ch. 8.

On the whole, we find that Monte Carlo methods have a number of attractive computational features that make them well-suited to PDE-based geometric computing. By avoiding the daunting challenge of mesh generation, these methods offer a framework that is scalable, parallelizable, easy-to-tune, and numerically robust (Fig. 1.1, *right*). They also occupy a unique place in the broader landscape of numerical solvers for PDEs, as they complement the strengths of existing grid-based techniques (*e.g.*, local versus global evaluation of solutions).

1.3 Related Work

The biggest issue preventing broader adoption of grid-free Monte Carlo methods like WoS (discussed in detail in Sec. 3.2) is that estimators have been developed only for a narrow set of PDEs beyond the original Laplace Dirichlet problem studied by Muller [174]—these include the Poisson and screened Poisson equations [51, 61], the heat equation [48, 95], the biharmonic equation [84, 154] and certain (mildly) nonlinear PDEs [28]. WoS has previously been used for specific problems in, *e.g.*, molecular dynamics [160], integrated circuit design [146], porous media [109], and electrostatics [108]. A few variants of WoS have also been developed, such as walking on rectangles [49, 216], on the boundary [217, 242], off-centered walks [110], and Green's function first passage (GFFP) methods [81, 107]; we draw on some of these techniques in this work.

Unfortunately, most prior work on grid-free Monte Carlo methods lacks thorough numerical evaluation, and even then only considers very simple geometry such as a box or cylinder. Moreover, questions essential for real applications like how to solve larger classes of equations, deal

with defective or highly detailed geometry, reduce variance and build high-performance systems have not been sufficiently explored. Despite some very close analogs in photorealistic rendering [147, 171], researchers and practitioners in high-performance computing also seem to be largely unaware of grid-free methods for PDEs. It is for this reason that we find this approach exciting: the computer graphics community has made Monte Carlo integration the workhorse of photorealistic rendering, and we believe that it can have a similar impact on PDE-based techniques which are central to computer-aided design and engineering.

Since our initial publication which introduced WoS to computer graphics [221], we have designed robust and efficient WoS estimators for a much broader set of fundamental PDEs [167, 223, 223, 224], which we will detail over the course of this text. Other graphics researchers have followed suit, and concurrently expanded the capabilities of WoS to, *e.g.*, support exterior and time-dependent problems [47, 179], simulate fluids and the coupled physics of radiation and diffusion [14, 211], solve inverse problems via differentiable formulations [272], and improve estimation quality through variance reduction [11, 151, 206].

1.4 Contribution

We develop a holistic Monte Carlo framework encompassing integration, variance reduction and accelerated geometric queries to solve linear elliptic PDEs in volumetric domains, *i.e.*, an N-dimensional solid region in $\mathbb{R}^N$ (we focus on $N = 2$ and 3). In particular, we provide:

1. a unified discussion of previously developed WoS estimators for elliptic PDEs (Ch. 3).
2. a new grid-free method called *walk on stars (WoSt)*, which generalizes WoS to solve PDEs with any arbitrary mix of Dirichlet, Neumann and Robin boundary conditions (Ch. 4)—these boundary conditions are a basic component of virtually every real physical system.
3. a new integral formulation and subsequent WoS estimators for PDEs with variable material coefficients, by establishing a close connection with *null-scattering* techniques [188] for rendering heterogeneous participating media (Ch. 5).
4. easy-to-implement estimators for spatial derivatives of PDE solutions (Ch. 3 & 5).
5. Efficient BVH-based geometric queries to accelerate the above estimators (Sec. 6.2).
6. several variance reduction strategies (Ch. 7), including:
    a) importance sampling of source terms, adaptive sampling of the solution, and control variates for spatial derivatives.
    b) a *boundary value caching (BVC)* scheme, similar in spirit to *virtual point light* methods in rendering [46, 132], that greatly amortizes the cost of long walks and suppresses the typical salt-and-pepper noise characteristic of independent Monte Carlo estimates.
    c) a *reverse* WoSt estimator that splats known boundary and source data to multiple points in the interior of a domain, by extending the recently developed *bidirectional* formulation of WoS [206] from PDEs with Dirichlet conditions to those with Neumann and Robin conditions as well.
    d) a *weight window* strategy from neutron transport [25, 102] that significantly reduces noise and improves efficiency in problems with high-frequency material coefficients.

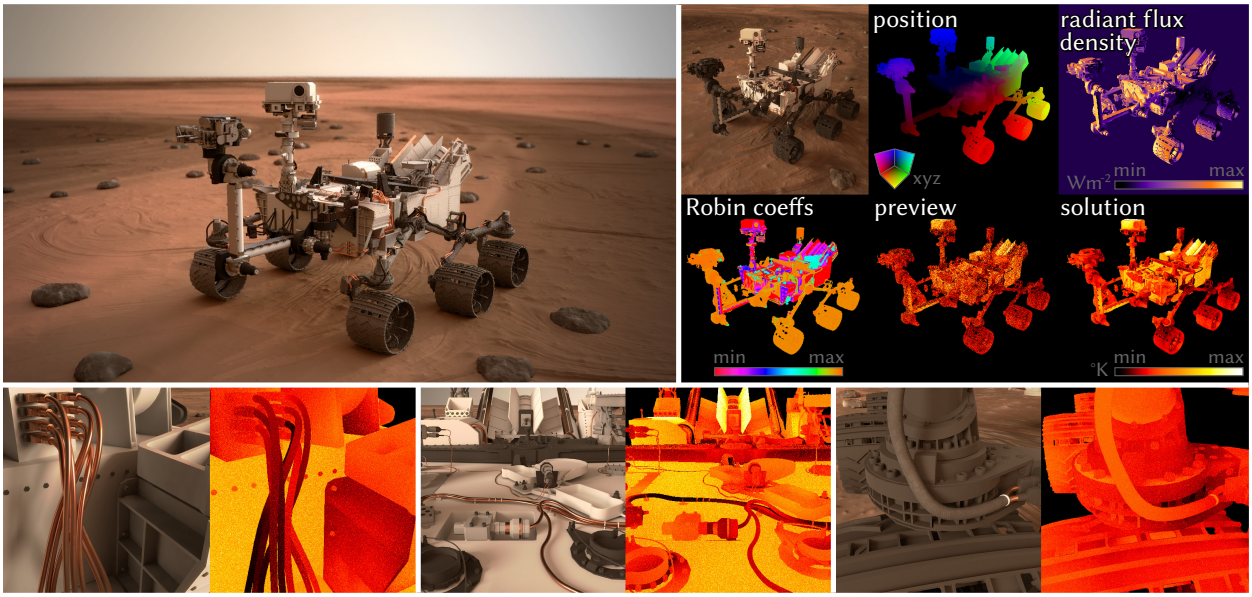


Figure 1.9: Thermal analysis of NASA's Curiosity Mars rover. Keeping temperatures within specified thermal limits is critical to mission success—but thermal modeling is historically difficult to integrate into the design phase, due to intricate geometry not easily captured via finite element models (Fig. 8.1). Here our walk on stars solver (Ch. 4) computes realistic temperature estimates quickly and progressively even for extremely complex geometry, without needing to volumetrically mesh the domain. A "deferred shading" approach provides output-sensitive evaluation, computing temperature values only at the points visible in screen space (top right). We can hence analyze temperature in local regions of interest, without computing a global solution (bottom row).

Our approach connects to a large body of work on Monte Carlo rendering [203]. From the PDE point of view, the major difference is that the differential equation governing radiative transfer is first order in space, whereas the PDEs we seek to solve have second order, diffusive terms that demand different numerical techniques. There are of course many parallels between these problems from a mathematical, computational, and system design point of view, which we explore throughout this text. More broadly, by framing PDE-based geometric computing in terms of stochastic processes, we build a bridge to rich tools not just from Monte Carlo rendering, but also reinforcement learning, stochastic control and mathematical finance. We evaluate the capabilities of our solvers in analyzing physical systems with complex geometry (see *e.g.*, Fig. 1.9) in Ch. 8, and end with a discussion on future directions in Ch. 9.

Chapter 2

Background: Differential & Integral Equations

We rely on concepts from the theory of partial differential equations, integral equations and stochastic calculus to develop Monte Carlo algorithms for solving PDEs. We provide essential background here as few readers will be familiar with all of these topics, and scope out the specific types of PDEs we will consider in this work. Chapter 3 then describes the Monte Carlo walk on spheres algorithm, which is the starting point for the generalizations of WoS we develop in Ch. 4 - 7 for solving these equations.

2.1 Partial Differential Equations

Many natural phenomena are described by relating rates of change. An *ordinary differential equation (ODE)* describes how a quantity changes in time. For instance, $d^2p(t)/dt^2 = -g$ implicitly describes the trajectory of a particle $p(t)$ under the influence of a gravitational acceleration g. Only by solving this ODE for $p(t)$ do we obtain an explicit description of the trajectory. Likewise, a PDE implicitly describes a function via relationships between partial derivatives in space. A prototypical example is the *Laplace equation* $d^2u(x,y,z)/dx^2 + d^2u(x,y,z)/dy^2 + d^2u(x,y,z)/dz^2 = 0$ (or more compactly $\Delta u = 0$), which describes the steady-state of a diffusion process, *i.e.*, the way heat diffuses smoothly from the domain boundary into the interior. Just like ODEs, we must ultimately solve for an explicit function u satisfying this relationship. Since most PDEs do not have analytical solutions (including the Laplace equation), we rely on numerical methods to solve them.

Given the vast number of PDEs and the diverse phenomena they model such as thermodynamics, electromagnetism, fluid mechanics, elasticity, quantum mechanics, and even the movement of stock prices, numerical methods are often specialized to consider specific classes of equations. PDEs can be classified based on their form and properties, as we describe below:

Order And Linearity. The *order* of a PDE refers to the highest-order degree of any derivative appearing in the PDE. For instance, the Laplace equation is 2nd order as it involves spatial derivatives no higher than degree two, whereas a *biharmonic equation* $\Delta^2 u = 0$ is 4th order. A PDE is *linear* if it is a linear polynomial in the function and its derivatives. The Laplace equation is linear, but the *inviscid Burger's equation* $\frac{\partial}{\partial t}u(x,t) = -u(x,t)\frac{\partial}{\partial x}u(x,t)$ is nonlinear as it multiplies the function by one of its derivatives. We will consider 2nd order linear PDEs in this text.

Ellipticity. Roughly speaking, *elliptic* equations are those whose solutions are captured by the idea of "repeated local averaging". These PDEs often describe steady-state processes, like the distribution of temperature in a stationary object. The Laplace equation is elliptic, as its solution at any point in a domain Ω equals the average value in some small neighborhood (*i.e.*, the *mean value property*, see Eq. 2.13). *Parabolic* PDEs instead describe time-dependent processes, like the changing temperature in an object over time. The canonical example is the heat equation $\Delta u(x,t) = \partial u(x,t)/\partial t$, whose solution u becomes progressively smoother and eventually reaches an

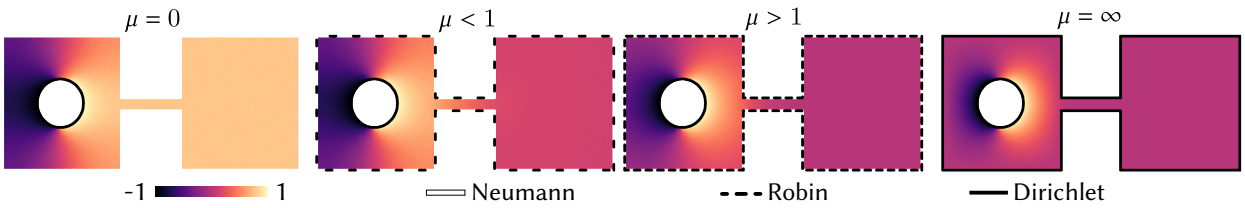


Figure 2.1: A non-negative Robin coefficient μ linearly interpolates between Dirichlet ($\mu = \infty$) and Neumann ($\mu = 0$) boundary conditions, which prescribe solution values and derivatives on the boundary, respectively.

equilibrium described by the Laplace equation as $t \to \infty$. *Hyperbolic* PDEs, on the other hand, are often associated with processes that involve wave propagation, such as sound waves.

We will focus on elliptic problems in this work, as their solutions can be computed by taking averages of local estimates via Monte Carlo integration (Sec. 3.1). Formally, a 2nd order linear PDE with constant coefficients is elliptic if its *principal symbol* (*i.e.*, the polynomial corresponding to its highest-order term) is greater than zero away from the origin. For PDEs with variable coefficients such as $\nabla \cdot (K(x)\nabla u) = 0$, ellipticity requires the diffusion matrix $K(x) : \Omega \mapsto \mathbb{R}^{N\times N}$ to be positive definite, *i.e.*, $\sum_{i,j=1}^{n} K^{ij}(x)\eta_i\eta_j > 0$ for all non-zero $\eta \in \mathbb{R}^N$ [64, 72].

Boundary Conditions. PDEs are often paired with an additional set of constraints called *boundary conditions*. Boundary conditions can be used to specify various physical constraints such as temperatures, voltages, forces and velocities on the boundary $\partial\Omega$ of a domain $\Omega \subset \mathbb{R}^N$, and typically have a major impact on the solution of a PDE. They come in many flavors: for instance with a Laplace equation, *Dirichlet* boundary conditions specify the solution value along $\partial\Omega$ (*e.g.*, a surface held at a fixed temperature), which uniquely determines the solution inside Ω. *Neumann* conditions instead specify the value of the normal derivative of the solution along $\partial\Omega$ (*e.g.*, the heat flux across a surface); the solution inside Ω is determined only up to an additive constant. *Robin* conditions linearly combine both solution values and their derivatives on $\partial\Omega$, which means that the PDE solution interpolates between the solution with pure Dirichlet conditions and pure Neumann conditions inside Ω (Fig. 2.1), and is uniquely determined. More generally, one can prescribe different boundary conditions on disjoint parts of $\partial\Omega$. For a Laplace equation, this yields a *boundary value problem (BVP)* of the form

$$\begin{aligned} \Delta u(x) &= 0 && \text{on } \Omega, \\ u(x) &= g(x) && \text{on } \partial\Omega_{\mathrm{D}}, \\ \tfrac{\partial u(x)}{\partial n_x} &= h(x) && \text{on } \partial\Omega_{\mathrm{N}}, \\ \tfrac{\partial u(x)}{\partial n_x} + \mu(x)u(x) &= \ell(x) && \text{on } \partial\Omega_{\mathrm{R}}, \end{aligned} \tag{2.1}$$

where the boundary is partitioned into a Dirichlet part $\partial\Omega_{\mathrm{D}}$ with prescribed values $g : \partial\Omega_{\mathrm{D}} \to \mathbb{R}$, a Neumann part $\partial\Omega_{\mathrm{N}}$ with prescribed derivatives $h : \partial\Omega_{\mathrm{N}} \to \mathbb{R}$, and a Robin part $\partial\Omega_{\mathrm{R}}$ with prescribed right hand side $\ell : \partial\Omega_{\mathrm{R}} \to \mathbb{R}$. Here n_x is the unit outward normal to $\partial\Omega$ at x, and $\mu \in \mathbb{R}_{\geq 0}$ is a non-negative Robin coefficient that can vary over $\partial\Omega_{\mathrm{R}}$ (negative values of μ occur less frequently in natural physical systems [86, Sec. 1], and we do not consider them here). Since Robin conditions serve as general first-order boundary conditions, we recover Neumann conditions when $\mu = 0$, and Dirichlet conditions as $\mu \to \infty$. A function u is *harmonic* if it satisfies Eq. 2.1 on Ω. We will provide Monte Carlo estimators for BVPs with any combination of Dirichlet, Neumann and Robin boundary conditions in Ch. 3 & 4.

Exterior Problems, Open Domains And Double-Sided Boundary Conditions. One may also solve PDEs in the exterior of a domain, *i.e.*, $\mathbb{R}^N \backslash \Omega$. The domain Ω need not be closed/watertight, and can have different boundary conditions prescribed on either side of $\partial\Omega$. We will describe how to handle such generalizations with our method in Sec. 3.2.4 and App. B.

2.1.1 Linear Elliptic Equations

Beyond a basic Laplace equation which essentially interpolates boundary values, 2nd order linear elliptic PDEs can also model rich spatially varying material properties of a medium. In thermodynamics, for example, PDEs with variable coefficients model how heterogeneous composite materials conduct or insulate heat—much as early algorithms for photorealistic rendering were motivated by predictive lighting design [144], such models can be used to predict and improve thermal efficiency in building design [276]. Likewise, variable permittivity in electrostatics impacts the design of antennas [194] and the simulation of biomolecules [66]; in hydrology, variable transmissivity of water through soil impacts remediation strategies for groundwater pollution [269]. More directly connected to our work, variable coefficients in the light transport equation are used to model heterogeneity in participating media [188]. Beyond spatially varying materials, variable coefficients can also be used to model curved geometry by using PDE coefficients on a flat domain to encode an alternative *Riemannian metric* (Fig. 8.8). Below, we discuss generalizations of the Laplace equation we will consider in this text:

Source term. Continuing with the heat analogy for a Laplace equation, a *source term* $f : \Omega \to \mathbb{R}$ adds additional "background temperature" to a PDE (Fig. 2.2, *center left*). For instance, a *Poisson equation* has the form

$$\Delta u(x) \quad = \quad -f(x) \quad \text{on } \Omega \tag{2.2}$$

subject to Dirichlet, Neumann or Robin boundary conditions.

Diffusion. The rate of diffusion in a spatially varying medium is modeled by replacing the operator Δ with $\nabla \cdot (\kappa(x)\nabla)$, where $\kappa : \Omega \to \mathbb{R}_{>0}$ is the *diffusion coefficient* (Fig. 2.2, *center*).

Drift. A *drift coefficient*, given by a vector field $\vec{\omega} : \Omega \to \mathbb{R}^N$, models the motion of a material in a particular direction. For instance, the *steady-state advection equation* $\vec{\omega}(x) \cdot \nabla u(x) = 0$ describes a quantity u that is unchanged as it flows along $\vec{\omega}$; adding this term to a Poisson equation causes heat to drift as it diffuses (Fig. 2.2, *center right*).

Absorption. An *absorption (or screening) coefficient* $\sigma : \Omega \to \mathbb{R}_{>0}$ models "cooling" of the solution due to the background medium; larger coefficient values dampen the solution more. *E.g.*, a *screened Poisson equation* (seen in Fig. 2.2, *far right*) is given by

$$\Delta u(x) - \sigma(x)u(x) \quad = \quad -f(x) \quad \text{on } \Omega, \tag{2.3}$$

again subject to boundary conditions.

Combining all these terms and coefficients then yields a linear elliptic equation of the form

$$\nabla \cdot (\kappa(x)\nabla u(x)) + \vec{\omega}(x) \cdot \nabla u(x) - \sigma(x)u(x) \quad = \quad -f(x) \quad \text{on } \Omega. \tag{2.4}$$

Here we do not require the source term $f : \Omega \to \mathbb{R}$ to be continuous, but we will assume the diffusion coefficient $\kappa : \Omega \to \mathbb{R}_{>0}$ is twice-differentiable, the drift coefficient $\vec{\omega} : \Omega \to \mathbb{R}^N$ is a

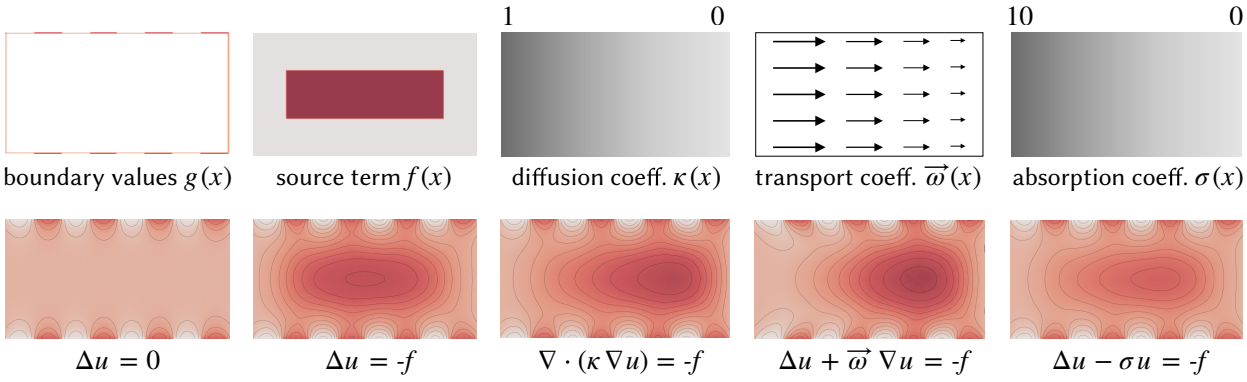


Figure 2.2: Effect of each spatially-varying term of Eq. 2.4 on the PDE solution.

vector field expressible as the gradient of a scalar field, and the absorption coefficient $\sigma : \Omega \to \mathbb{R}_{\geq 0}$ is continuous (C^0). Though these coefficients can be generalized further (for instance, κ can be any arbitrary positive definite matrix and σ can be negative), the conditions we impose on them are sufficient to ensure ellipticity [64, 72]. Moreover, these conditions will enable us to derive Monte Carlo estimators for variable-coefficient equations in Ch. 5 that do not have to resort to *numerical homogenization* [52], and can directly resolve the original, detailed solution (see, *e.g.*, Fig. 5.1).

2.1.2 Fundamental Solutions

A *Green's function* captures the influence of the source term f on the solution of a linear elliptic equation *with constant coefficients*. In particular, it describes the (fundamental) solution when the source is a Dirac delta distribution δ_x centered at a single point $x \in \Omega$. For instance, in the case of Eq. 2.2, the Green's function $G^\Omega(x, y)$ is the solution to the Poisson equation $\Delta u(y) = -\delta_x(y)$. In general, Green's functions will depend on the shape of the domain Ω and the choice of boundary conditions—as a result, they are typically not known in closed-form. However, explicit expressions are available for important special cases, *e.g.*, the *free space* Green's function $G^{\mathbb{R}^N}$ on $\Omega = \mathbb{R}^N$, and the Green's function G^B for a ball $\Omega = B$ with zero-Dirichlet boundary conditions (App. A). The walk on spheres and walk on stars algorithms we describe in the following chapters will effectively provide a bridge between closed-form Green's functions on special domains, and solutions to PDEs on more general domains.

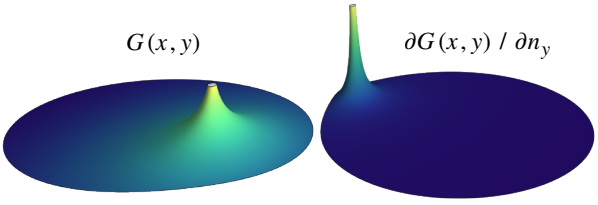


Figure 2.3: The Green's function and its normal derivative are singular at the point they are centered on, but decay smoothly and fall-off quickly away from the singularity.

The *Poisson kernel* likewise captures the influence of the boundary conditions on the solution, *e.g.*, when the Dirichlet function g is a Dirac delta distribution δ_x centered on a single boundary point $x \in \partial\Omega$. At any point $y \in \Omega$ with associated normal n_y, it can be expressed as the normal derivative of a Green's function:

$$P^\Omega(x, y) := -\frac{\partial G^\Omega(x, y)}{\partial n_y}. \tag{2.5}$$

As with Green's functions, common Poisson kernels are known explicitly in $\mathbb{R}^N$ and for a ball.

2.2 Boundary Integral Equation

One can often reformulate linear elliptic PDEs as *recursive integral equations*, akin to the classic *rendering equation* [121]. As in rendering, these equations can be solved without discretizing space, by recursively applying Monte Carlo integration (as will be discussed in Ch. 3). In this section, we provide a *boundary integral* formulation for PDEs with constant coefficients (*e.g.*, Eq. 2.2). Section 2.3 then describes a *stochatic integral* formulation for Eq. 2.4 with varying coefficients.

Derivation. The solution u to a Poisson equation $\Delta u = -f$ can be expressed via an integral involving the associated Green's function and Poisson kernel. Assume for now that the domain Ω is watertight with smooth boundary $\partial\Omega$, and let x be an evaluation point in the interior of Ω. We first multiply the Poisson equation with its Green's function G^Ω, and integrate over Ω to get

$$0 = \int_\Omega G^\Omega(x,y)\ \Delta u(y)\,\mathrm{d}y\ +\ \int_\Omega G^\Omega(x,y)\ f(y)\,\mathrm{d}y. \tag{2.6}$$

Applying integration by parts to the first integral, we have

$$0 = \int_{\partial\Omega} G^\Omega(x,z)\ \frac{\partial u(z)}{\partial n_z}\,\mathrm{d}z\ -\ \int_\Omega \nabla G^\Omega(x,y)\ \cdot \nabla u(y)\,\mathrm{d}y\ +\ \int_\Omega G^\Omega(x,y)\ f(y)\,\mathrm{d}y. \tag{2.7}$$

Applying integration by parts again to the second integral, and rearranging terms then yields

$$\begin{aligned}\int_\Omega u(y)\ \Delta G^\Omega(x,y)\,\mathrm{d}y = &\int_{\partial\Omega} \frac{\partial G^\Omega(x,z)}{\partial n_z}\ u(z)\ -\ G^\Omega(x,z)\ \frac{\partial u(z)}{\partial n_z}\,\mathrm{d}z \\ &- \int_\Omega G^\Omega(x,y)\ f(y)\,\mathrm{d}y.\end{aligned} \tag{2.8}$$

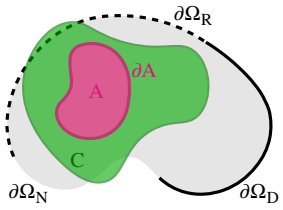
From the definitions $\Delta G^\Omega(x,y) = -\delta_x^\Omega(y)$ and $P^\Omega(x,z) = {}^{-\partial G^\Omega(x,z)}/_{\partial n_z}$, we arrive at

$$\begin{aligned}u(x) = &\int_{\partial\Omega} P^\Omega(x,z)\ u(z)\ +\ G^\Omega(x,z)\ \frac{\partial u(z)}{\partial n_z}\,\mathrm{d}z \\ &+ \int_\Omega G^\Omega(x,y)\ f(y)\,\mathrm{d}y.\end{aligned} \tag{2.9}$$

This equation determines the solution u at x entirely through the solution values $u(z)$ and normal derivatives ${}^{\partial u(z)}/_{\partial n_z}$ on the boundary $\partial\Omega$, and the source values $f(y)$ inside the domain Ω. From Eq. 2.1, the Dirichlet, Neumann and Robin parts of the boundary have prescribed values g, h and k, respectively, while f is specified inside the domain for a Poisson equation. To use Eq. 2.9, we must then determine unknown solution values $u(z)$ on the Neumann boundary $\partial\Omega_\mathrm{N}$, unknown derivative values ${}^{\partial u(z)}/_{\partial n_z}$ on the Dirichlet boundary $\partial\Omega_\mathrm{D}$, and either $u(z)$ and ${}^{\partial u(z)}/_{\partial n_z}$ on the Robin boundary $\partial\Omega_\mathrm{R}$, via the relation ${}^{\partial u}/_{\partial n} + \mu u = \ell$ when $\mu > 0$.

2.2.1 General Setting

In practice, Eq. 2.9 cannot be used directly since the Green's function and Poisson kernel for an arbitrary domain Ω are unknown. Fortunately, this equation can be generalized to the *boundary integral equation (BIE)* [41, Section 2] where these functions are no longer tied to the domain Ω. Instead one may use, *e.g.*, the closed-form Green's function and Poisson kernel for a ball or for $\mathbb{R}^N$. Moreover, while we ultimately seek a solution on Ω, the BIE applies to arbitrary subdomains in Ω:



Boundary Integral Equation

For any two sets $\mathrm{A} \subset \Omega$ and $\mathrm{C} \subset \mathbb{R}^N$, and for any point $x \in \mathbb{R}^N$, the solution to a Poisson equation satisfies

$$\alpha(x)\, u(x) = \int_{\partial \mathrm{A}} P^{\mathrm{C}}(x,z)\, u(z) \;+\; G^{\mathrm{C}}(x,z)\, \frac{\partial u(z)}{\partial n_z}\, \mathrm{d}z + \int_{\mathrm{A}} G^{\mathrm{C}}(x,y)\, f(y)\, \mathrm{d}y, \tag{2.10}$$

where

$$\alpha(x) := \begin{cases} 1, & x \in \mathrm{A}, \\ 1/2, & x \in \partial \mathrm{A}, \\ 0, & x \notin \mathrm{A}. \end{cases} \tag{2.11}$$

Hunter and Pullan [106, Chapter 3.3] provide a derivation. We note that if $\partial \mathrm{A}$ is a non-smooth curve in the plane, then $\alpha = 1 - \theta/2\pi$ at a corner with interior angle θ. To keep things simple, we will assume $\partial \mathrm{A}$ is smooth, letting $\alpha = 1/2$ at all boundary points.

Though we focus on Poisson equations for simplicity, the BIE extends immediately to *screened Poisson equations* $\Delta u - \sigma u = -f$ with constant absorption coefficient $\sigma \in \mathbb{R}_{\geq 0}$: the only modification to Eq. 2.10 is to replace the Green's function and Poisson kernel with their screened counterparts (App. A.2). Likewise, for a constant diffusion coefficient $\kappa \in \mathbb{R}_{>0}$, the Green's function simply scales by a factor κ. Constant drift along a fixed direction $\vec{\omega} \in \mathbb{R}^N$ can be captured via the *von Mises–Fisher distribution* [75, 215]. Boundary integral equations are also readily available for a variety of other PDEs not directly considered in this work, such as the Helmholtz equation [106, Chapter 3], linear elasticity [106, Chapter 4] and the biharmonic equation [111]. Below, we discuss a few special cases of Eq. 2.10 with different choices for the sets A and C, and describe how the BIE can be generalized to support double-sided boundary conditions in closed and open domains.

Boundary Element Formulation

Conventional numerical solvers like the boundary element method integrate Eq. 2.10 over the PDE domain ($\mathrm{A} = \Omega$) using free-space kernels ($\mathrm{C} = \mathbb{R}^N$). BEM does not directly support source terms f, leading to the integral

$$\alpha(x)\, u(x) = \int_{\partial \Omega} P^{\mathbb{R}^N}(x,z)\, u(z) \;+\; G^{\mathbb{R}^N}(x,z)\, \frac{\partial u(z)}{\partial n_z}\, \mathrm{d}z. \tag{2.12}$$

To determine the unknown data u and $\partial u / \partial n$ on $\partial \Omega$, BEM uses a finite basis of functions (associated with mesh nodes on a discretized boundary) to solve a dense linear system—resulting in the tradeoffs discussed in Sec. 8.2.1.

Mean Value Property Of Harmonic Functions

Monte Carlo methods like walk on spheres (Sec. 3.2) instead integrate the BIE over a ball $\mathrm{B}(x, R) \subset \Omega$ of radius R centered at x, adopting kernels from the ball ($\mathrm{A} = \mathrm{C} = \mathrm{B}(x, R)$). At points $z \in \partial \mathrm{B}$, these kernels then simplify to $G^{\mathrm{B}}(x,z) = 0$ and $P^{\mathrm{B}}(x,z) = 1/|\partial \mathrm{B}|$ (*i.e.*, 1 over the surface area of the ball boundary), yielding the mean value property of harmonic functions

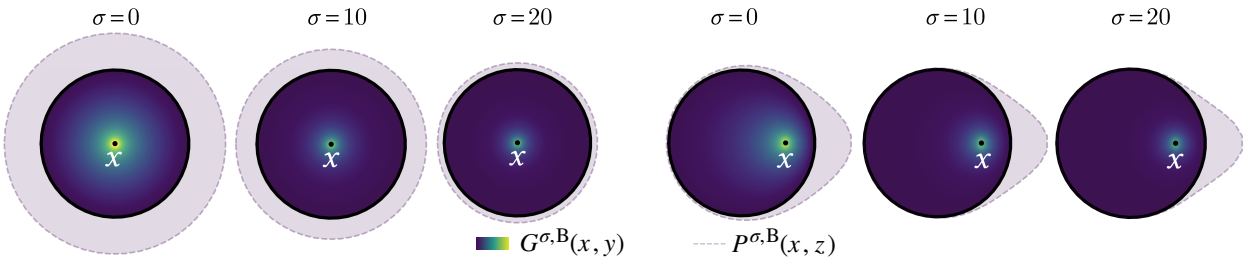


Figure 2.4: Left: As σ is increased, the Green's function $G^{\sigma,\mathrm{B}}(x,y)$ for a screened Poisson equation becomes more localized around the point x, and the magnitude of the Poisson kernel $P^{\sigma,\mathrm{B}}(x,z)$ shrinks. These functions revert to the harmonic Green's function and Poisson kernel as $\sigma \to 0$. Right: The functions $G^{\sigma,\mathrm{B}}$ and $P^{\sigma,\mathrm{B}}$ are not rotationally symmetric when the point x does not coincide with the center of the ball.

when $f = 0$:

$$u(x) = \frac{1}{|\partial \mathrm{B}(x,R)|} \int_{\partial \mathrm{B}(x,R)} u(z)\, \mathrm{d}z \tag{2.13}$$

This setup greatly simplifies the BIE by eliminating dependence on $\partial u/\partial n$. Unlike BEM, a non-zero source term f is accounted for by adding the integral

$$\int_{\mathrm{B}(x,R)} G^{\mathrm{B}}(x,y)\; f(y)\, \mathrm{d}y. \tag{2.14}$$

More importantly, WoS evaluates 2.13 by recursively estimating $u(z)$ on $\partial \mathrm{B}$. It therefore does not need to discretize the domain Ω or its boundary $\partial\Omega$, nor solve a global system of equations.

Off-Centered Mean Value Property

The point of evaluation x in Eq. 2.13 need not coincide with the center $c \in \Omega$ of a ball $\mathrm{B}(c,R)$. With a screened Poisson equation, this leads to the following more general *off-centered* formulation [55, 110] of the mean value property:

$$u(x) = \int_{\partial \mathrm{B}(c,R)} P^{\sigma,\mathrm{B}}(x,z)\; u(z)\, \mathrm{d}z \;+\; \int_{\mathrm{B}(c,R)} G^{\sigma,\mathrm{B}}(x,y)\; f(y)\, \mathrm{d}y. \tag{2.15}$$

We provide explicit expressions for off-centered versions of the functions $G^{\sigma,\mathrm{B}}$ and $P^{\sigma,\mathrm{B}}$ in App. A.2.2. As shown in Fig. 2.4, the Poisson kernel $P^{\sigma,\mathrm{B}}(x,z)$ reduces to $1/|\partial \mathrm{B}(x,R)|$ when $x = c$ and $\sigma = 0$, recovering the usual mean value property. In Sec. 5.2.2, we will use Eq. 2.15 to design a WoS estimator for solving variable-coefficient PDEs.

Double-Sided Boundary Conditions

The BIE for double-sided boundary conditions in an open domain $\Omega \subset \mathbb{R}^N$ is given by [41]

$$\begin{aligned} \alpha(x)\; u(x) = &\int_{\partial\Omega} P^{+}(x,z)\left[u^{+}(z) - u^{-}(z)\right] \;+\; G(x,z)\left[\frac{\partial u^{+}(z)}{\partial n_z^{+}} - \frac{\partial u^{-}(z)}{\partial n_z^{-}}\right] \mathrm{d}z \\ &+ \int_{\Omega} G(x,y)\; f(y)\, \mathrm{d}y, \end{aligned} \tag{2.16}$$

where n^{+} and n^{-} denote unit normals on either side of $\partial\Omega$ (respectively), u^{+} and u^{-} represent corresponding solution values on $\partial\Omega$, and $P^{+}(x,z) := -\partial G(x,z)/\partial n_z^{+}$. Since all points are either on the boundary or the domain interior, $\alpha = 1/2$ on $\partial\Omega$ and 1 otherwise. In App. B, we discuss how to apply the walk on stars algorithm for BVPs with Dirichlet, Neumann and Robin boundary conditions (Ch. 4) to open domains and double-sided boundaries.

2.3 Feynman–Kac Formula

In general, we do not have boundary integral representations of PDEs with variable coefficients (like Eq. 2.4), due to the unavailability of Green's functions in $\mathbb{R}^N$, on a ball, or elsewhere. However, the solution to an elliptic PDE can also be described in terms of continuous stochastic processes such as *Brownian motion*, via the *Feynman–Kac formula* from stochastic calculus [191, Ch. 8]. This formula will provide a critical starting point in Ch. 5 for solving variable-coefficient equations, as it is more general than the BIE from the previous section. Moreover, it has close parallels with volume rendering, providing us with key techniques for numerical integration.

Here we provide essential background on stochastic processes (Sec. 2.3.1) and their associated integral representations of PDEs (Sec. 2.3.2), which at present are not widely used in computer graphics (we refer to Øksendal [191] for a more comprehensive introduction). In particular, the central object that we would like to simulate is Brownian motion, as it captures the phenomenon of *diffusion* described by elliptic equations through the Feynman–Kac formula. Unfortunately, simulating Brownian motion directly is both expensive and introduces statistical error, especially in domains with complex geometry. We will see later that walk on spheres and its generalizations (Ch. 3 - 5) enable a far more efficient simulation with significantly less statistical error.

2.3.1 Stochastic Processes

A stochastic process is a collection of random variables X_t that represents the evolution of a system over time $t \geq 0$. The process is continuous if it can be observed continuously with time. A key characteristic of a stochastic process is that it incorporates some form of randomness or unpredictability, which means that even if the initial state $X_0 = x$ is known, the future evolution of a process cannot be predicted with certainty. Such a process is therefore typically modelled by a *probability density function (PDF)* that is non-negative everywhere, and integrates to 1 over the domain on which it is defined. We can use a PDF to calculate the probability with which future random states $X_{t>0}$ of the process take on a permissible range of values. In particular, for a real-valued random variable X_t, integrating the PDF p over an arbitrary interval $[a, b]$ gives the probability that X_t lies inside the interval:

$$\mathbb{P}\{a \leq X_t \leq b\} = \int_a^b p(x)\,\mathrm{d}x. \tag{2.17}$$

A *cumulative density function (CDF)* $P(x)$ represents the probability that X_t takes on a value less than or equal to x, *i.e.*, $P(x) := \mathbb{P}\{X_t \leq x\}$.

For our purposes, we will consider a continuous time-parameterized family of $\mathbb{R}^N$-valued random variables X_t on a domain A with PDF p^{A}. The expected value—or mean of all possible values—of any L^1-integrable function $\phi : \mathrm{A} \to \mathbb{R}$ is then given by

$$\mathbb{E}\left[\phi(X_t)\right] = \int_{\mathrm{A}} \phi(x)\; p^{\mathrm{A}}(x)\,\mathrm{d}x. \tag{2.18}$$

Throughout, we will informally refer to a realization of a stochastic process as a *random walk*.

Brownian Motion

The central example of a continuous stochastic process is Brownian motion, which is more formally known as a *Wiener process*. Intuitively, a Wiener process describes a random walk by repeatedly taking small Gaussian steps, and letting the variance of the Gaussian go to zero (Fig.

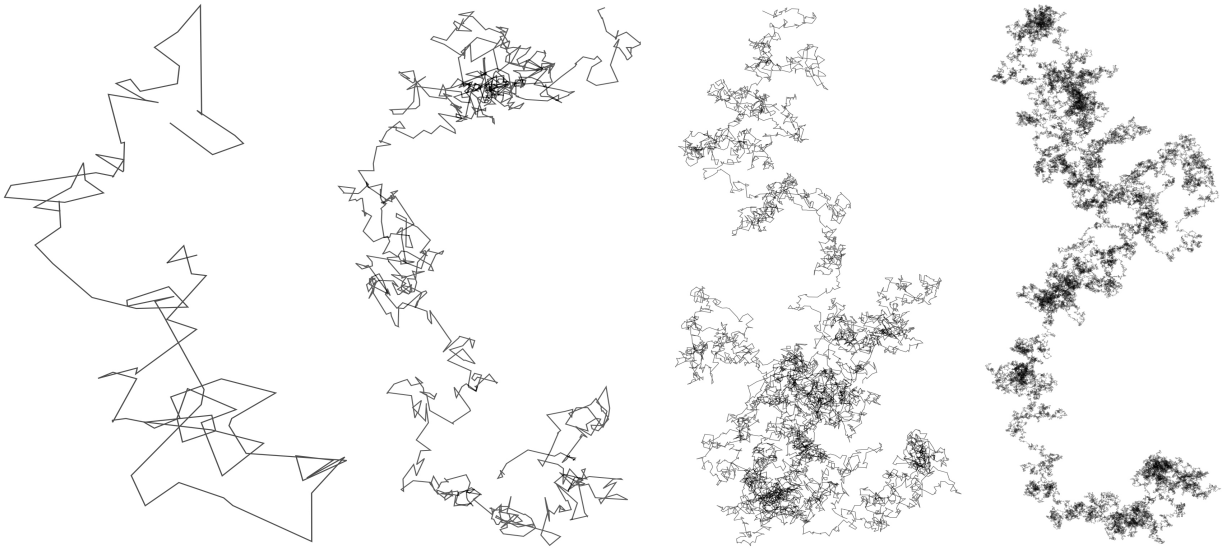


Figure 2.5: Brownian motion can be thought of as the limit of taking small, normally distributed steps.

2.5). This motion is hence *isotropic*, *i.e.*, there is no preferred direction, and it has no "memory", *i.e.*, the direction of motion at each point in time is independent of all past motion. More formally, a (multidimensional) Wiener process starting at a point $x_0 \in \mathbb{R}^N$ is a time-parameterized family of $\mathbb{R}^N$-valued random variables W_t characterized by the following criteria:

- $W_0 = x_0$
- The process has Gaussian increments, *i.e.*, for all $t, s \geq 0$, the increments $W_{t+s} - W_t \sim \mathcal{N}(0, s)$ are normally distributed with mean 0 and variance s.
- These increments are independent of all previous random variables W_r for $0 \leq r \leq t$.
- The process is almost surely continuous with respect to t, *i.e.*, random walks have continuous paths in $\mathbb{R}^N$.

Diffusion Process

Just as Brownian motion will help us model isotropic diffusion, we will use more general stochastic processes to model diffusion that is anisotropic or exhibits *drift* in a particular direction (Fig. 2.6, *center*). Such processes can be defined by combining a deterministic velocity with stochastic "noise", modeled via Brownian motion. In particular, the *stochastic differential equation (SDE)*

$$\mathrm{d}X_t = \vec{\omega}(X_t)\,\mathrm{d}t + \mathrm{d}W_t \tag{2.19}$$

describes a stochastic process whose increments $\mathrm{d}X_t$ behave exactly like Brownian increments $\mathrm{d}W_t$, offset by a deterministic vector $\vec{\omega}$ (for a more formal treatment of the notation $\mathrm{d}X_t$, see Øksendal [191, Ch. 5]). More generally, a *diffusion process* is any stochastic process of the form

$$\mathrm{d}X_t = \vec{\omega}(X_t)\,\mathrm{d}t + K(X_t)\,\mathrm{d}W_t, \tag{2.20}$$

where for each time t and location X_t, $\vec{\omega}(X_t) \in \mathbb{R}^N$ gives a direction of drift, and $K(X_t)$ is a symmetric positive definite $N \times N$ matrix that controls the rate and directional bias of diffusion

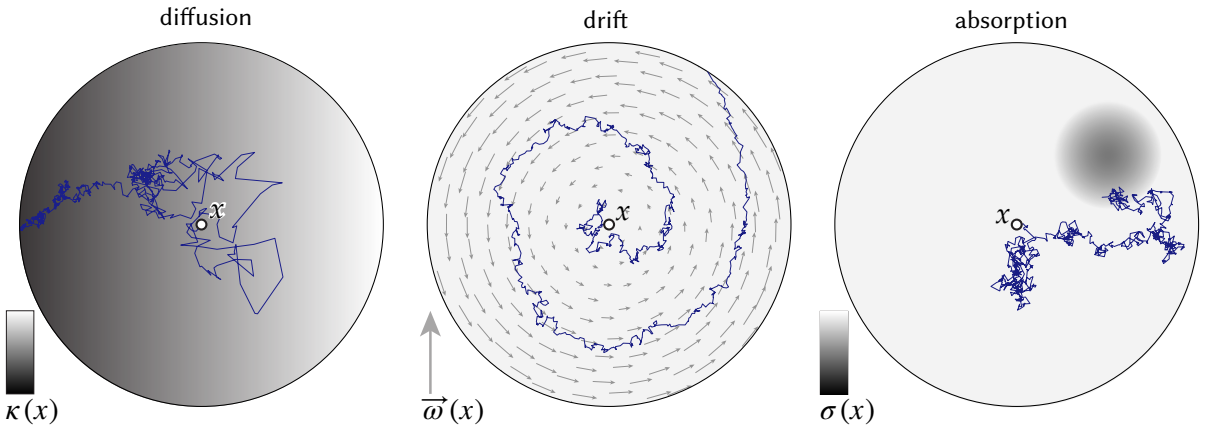


Figure 2.6: Components of a diffusion process. Note that a diffusion process starting at x will not in general have a uniform exit distribution over a sphere, as it is not isotropic like Brownian motion. Left: the diffusion coefficient $\kappa(x)$ modulates the size of random increments. Center: the drift coefficient $\vec{\omega}(x)$ adds deterministic offsets to the trajectory. Right: the screening coefficient $\sigma(x) > 0$ describes the probability of a random walk being absorbed, whereas $\sigma(x) < 0$ essentially describes emission of new random walks (which we do not consider in this work).

(*i.e.*, the variance and covariance of the random increments). In our setting, we will typically consider a simpler special case, namely

$$\mathrm{d}X_t = \vec{\omega}(X_t)\,\mathrm{d}t + \sqrt{\kappa(X_t)}\,\mathrm{d}W_t, \tag{2.21}$$

where $\kappa : \mathbb{R}^N \to \mathbb{R}_{>0}$ is a scalar- rather than matrix-valued function that determines the standard deviation of the normal distribution associated with W_t (Fig. 2.6, *left*). We assume that neither $\vec{\omega}$ or κ depend on time.

Killed Diffusion Process

We can enrich our model of stochastic processes by also allowing a diffusion process to be probabilistically *killed*—intuitively, a random walk realized from such a process can be absorbed into the background medium (Fig. 2.6, *right*). This process will be essential for making the connection to PDEs with an absorption coefficient $\sigma \in \mathbb{R}_{\geq 0}$. We assume that the probability with which a random walk is absorbed is exponentially decaying for any value of σ. Then $p(t) = \sigma \mathrm{e}^{-\sigma t}$ gives the probability density of a killed diffusion process, and for a total time T, the corresponding probability is [203, Sec. 13.3.1]

$$\int_0^T \sigma \mathrm{e}^{-\sigma t}\,\mathrm{d}t = 1 - \mathrm{e}^{-\sigma T}. \tag{2.22}$$

Here a larger value of σ yields a higher probability of absorption.

Restriction To Bounded Domains

So far we've assumed that a diffusion process X_t is free to wander around all of $\mathbb{R}^N$, but suppose we are interested in a particular domain $\Omega \subset \mathbb{R}^N$. A common question we might ask is, "where does X_t first exit the domain Ω?" In other words, what is the probability that X_t first hits any given point on the boundary $\partial\Omega$. This information will be central to our approach to solving PDEs, as it corresponds to the Poisson kernel discussed in Sec. 2.1.2.

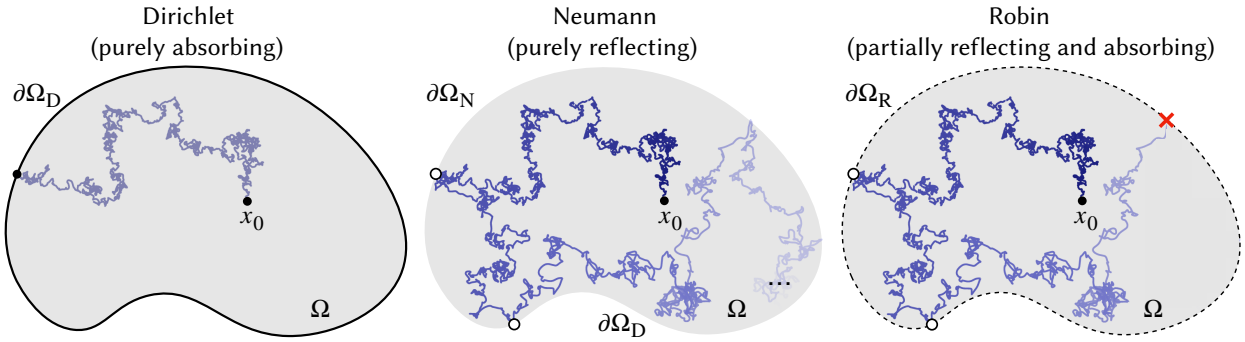


Figure 2.7: A random walk terminates when it hits an absorbing Dirichlet boundary $\partial\Omega_D$ (left), but is pushed back into the domain along the normal to a reflecting Neumann $\partial\Omega_N$ (center) or Robin boundary $\partial\Omega_R$ (right). The walk continues forever with $\partial\Omega_N$, but is eventually absorbed on $\partial\Omega_R$.

In the study of stochastic processes [27], the presence of a boundary $\partial\Omega$ is often described by using another random variable τ called *stopping time*, which is a path-dependent quantity describing when a diffusion process starting at $x \in \Omega$ first hits the boundary, *i.e.*, $\tau := \inf\{t \geq 0 : x + X_t \in \partial\Omega\}$. Then, the corresponding exit location X_τ on $\partial\Omega$ is a random variable as well, and the probability density of an associated random trajectory of the process in Ω is given by the Poisson kernel P^Ω [191, Ch. 7]. One could also consider the distribution of exit times on $\mathbb{R}_{\geq 0}$: though the exit time density is well-defined for time-independent elliptic PDEs [68, 95], it is generally not needed to estimate their solution.

As we will discuss in Sec. 2.3.2, PDEs with Dirichlet boundary conditions require simulating random walks that are stopped when they first hit $\partial\Omega$ (Fig. 2.7, *left*). Hence from the perspective of stochastic processes, Dirichlet conditions model absorbing boundaries. However, a random walk need not always stop when it hits the boundary—as shown in Fig. 2.7, Neumann and Robin boundary conditions instead require random walks to reflect off $\partial\Omega$ in the normal direction [86, 87]. Walks eventually stop on Robin boundaries, as they are both reflecting *and* absorbing.

Discretized Random Walks

The standard approach for simulating a diffusion process in $\mathbb{R}^N$ is to use explicit time stepping [101, 134, 172], akin to ray marching [248] or forward Euler. The *Euler-Maruyama method*, for instance, uses the following update rule to integrate the SDE in Eq. 2.21 with time step $h > 0$:

$$X_{k+1} = X_k + \vec{\omega}(X_k)h + \sqrt{\kappa(X_k)}\,(W_{k+1} - W_k)\,, \quad W_{k+1} - W_k \sim \mathcal{N}(0, h) \tag{2.23}$$

Unfortunately, this approach introduces several sources of error in bounded domains. *E.g.*, random walks can easily leave a domain Ω and must be clamped to the boundary (Fig. 2.8); shrinking h reduces discretization error, but significantly slows down computation from needing to take many small steps inside Ω. Error is exacerbated in problems with variable diffusion and drift coefficients, which implicitly modify the ideal step size. Though there exist integration schemes with better convergence properties [42, 148, 155, 168], SDE integrators are fundamentally not well-suited for simulating continuous random processes in bounded domains—in Sec. 8.3.1, we will demonstrate that techniques based on WoS offer a much more favorable runtime-to-bias tradeoff for both

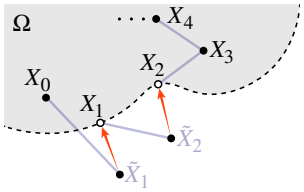


Figure 2.8: Discretized random walks can leave the domain, biasing results.

boundary-dominated and variable-coefficient problems, as they exactly model large steps of a Brownian random walk.

2.3.2 Stochastic Representation of PDEs

Notice that parameters κ, $\vec{\omega}$ and σ of a diffusion process resemble the coefficients of the PDEs from Sec. 2.1.1. The Feynman–Kac formula makes this relationship explicit by expressing the solution to Eq. 2.4 as an expectation over random trajectories of X_t. We refer to Øksendal [191, Sec. 8.2] for a derivation of this result (via a fundamental tool in stochastic calculus called *Ito's lemma* [191, Ch. 4]). Here we build up to the Feynman–Kac formula by starting with the important special case of *Kakutani's principle*, which effectively tells us how to solve a Laplace equation with Dirichlet boundary conditions by taking many random walks to the boundary.

Kakutani's Principle

Consider the Laplace equation

$$\begin{aligned} \Delta u(x) &= 0 \quad \text{on } \Omega, \\ u(x) &= g(x) \quad \text{on } \partial\Omega \end{aligned} \tag{2.24}$$

on a domain $\Omega \subset \mathbb{R}^N$, and let W_t be a Brownian process starting at a point $W_0 = x \in \Omega$. In this case, Kakutani's principle [122] states that

$$u(x) = \mathbb{E}[g(W_\tau)], \tag{2.25}$$

where as before τ is the (random) time when W_t first hits the domain boundary $\partial\Omega$. In other words, the solution to a Laplace equation is just the average boundary value "seen" by random walks starting at x.

Connection To Mean Value Property. Kakutani's principle can be viewed as a generalization of the mean value property (Eq. 2.13), since in the special case where the domain Ω is a ball $\mathrm{B}(x, R)$ of (any) radius R centered at the start of a random walk, the exit distribution of Brownian motion is uniform over the boundary sphere $\partial\mathrm{B}$ (*i.e.*, it equals $1/|\partial\mathrm{B}|$). Hence, Kakutani's principle in this case gives just the mean value integral

$$u(x) = \mathbb{E}[g(W_\tau)] = \frac{1}{|\partial\mathrm{B}(x,R)|} \int_{\partial\mathrm{B}(x,R)} g(z)\,\mathrm{d}z. \tag{2.26}$$

Source Term

For PDEs with a source term f, such as the Poisson equation $\Delta u = -f$, the solution u additionally picks up a term capturing the average heat "felt" by a random walk along its path [191, Ch. 9]:

$$\mathbb{E}\left[\int_0^\tau f(W_t)\,\mathrm{d}t\right]. \tag{2.27}$$

Connection To Green's function. The solution to a Poisson equation can also be expressed by convolving f with the Green's function G^Ω of the domain, if we assume the domain boundary $\partial\Omega$ only has zero Dirichlet conditions ($g = 0$), and $G^\Omega = 0$ on $\partial\Omega$. In this case, the BIE in Eq. 2.9 simplifies to

$$u(x) = \int_\Omega G^\Omega(x,y)\; f(y)\,\mathrm{d}y. \tag{2.28}$$

Comparing against Eq. 2.27, we thus have

$$\mathbb{E}\left[\int_0^\tau f(W_t)\,\mathrm{d}t\right] = \int_\Omega G^\Omega(x,y)\; f(y)\,\mathrm{d}y, \tag{2.29}$$

which implies that the Green's function in fact describes the locations where a Brownian random walk is likely to spend time within the domain Ω [191, Sec 9.2].

Screening Term

To model the effect of absorption, as in a screened Poisson equation $\Delta u - \sigma u = -f$, we incorporate the absorption coefficient $\sigma \in \mathbb{R}_{\geq 0}$ into the boundary and source terms to get

$$\mathbb{E}\left[\mathrm{e}^{-\sigma\tau} g(W_\tau)\right] \qquad \text{and} \qquad \mathbb{E}\left[\int_0^\tau \mathrm{e}^{-\sigma t} f(W_t)\,\mathrm{d}t\right], \tag{2.30}$$

respectively [191, Ch. 8]. Notice that *larger* values of σ yield *smaller* contributions. This exponential downweighting of the solution is accounted for in the boundary integral equation for a screened Poisson equation through its Green's function and Poisson kernel (App. A.2). For a spatially varying coefficient $\sigma(x)$, we simply replace $-\sigma\tau$ with $-\int_0^\tau \sigma(W_t)\,\mathrm{d}t$ in both terms.

Feynman–Kac

Finally, to account for spatially varying diffusion $\kappa(x) \in \mathbb{R}_{>0}$ and drift $\vec{\omega}(x) \in \mathbb{R}^N$, we replace the Brownian motion W_t with a general diffusion process X_t from Eq. 2.21. Combining expressions for the boundary, source, and absorption terms from Eqs. 2.25–2.30, we then arrive at:

The Feynman–Kac formula

For any point $x \in \Omega$, the solution to Eq. 2.4 with Dirichlet boundary conditions satisfies

$$u(x) = \mathbb{E}\left[\mathrm{e}^{-\int_0^\tau \sigma(X_t)\,\mathrm{d}t} g(X_\tau) \;+\; \int_0^\tau \mathrm{e}^{-\int_0^t \sigma(X_s)\,\mathrm{d}s} f(X_t)\,\mathrm{d}t\right]. \tag{2.31}$$

With Dirichlet conditions, a random walk terminates when $X_\tau \in \partial\Omega_\mathrm{D}$. With Neumann and Robin conditions, the primary change to the Feynman–Kac formula is that X_t must be reflected off $\partial\Omega_\mathrm{N}$ and $\partial\Omega_\mathrm{R}$ (Fig. 2.7). We refer to Morillon [172] for details on how to augment Eq. 2.31 with contributions from non-zero Neumann and Robin data h and ℓ respectively (Eq. 2.1).

Direct Estimation Using Discretized Random Walks. To approximately compute the solution to an elliptic PDE, one can use, *e.g.*, Euler-Maruyama with step-size h to simulate M random walks in Ω starting from x (Eq. 2.23). We can then estimate each walk's contribution by discretizing Eq. 2.31, and averaging results as follows:

$$\frac{1}{M}\sum_{i=1}^{M} Y_i, \quad \text{where } Y_i := \mathrm{e}^{-h\cdot\sum_{k=0}^{N-1}\sigma(X_k)} g(X_N) \;+\; h\cdot\sum_{k=0}^{N-1} \mathrm{e}^{-h\cdot\sum_{l=0}^{k-1}\sigma(X_l)} f(X_k). \tag{2.32}$$

Here N is the number of steps a random walk takes before it terminates on $\partial\Omega_\mathrm{D}$. In addition to the inefficiencies of discretized walks described in Sec. 2.3.1, a direct summation of this kind introduces further discretization error into the solution. Error also arises from naive estimation of the function $\exp(-\int_0^\tau \sigma(X_t)\,\mathrm{d}t)$, since nonlinear functions ϕ do not in general commute with expectations ($\mathbb{E}[\phi(X)] \neq \phi(\mathbb{E}[X])$). We briefly turn to volume rendering next, as it will inspire techniques for solving variable-coefficient PDEs with WoS that avoid discretization entirely.

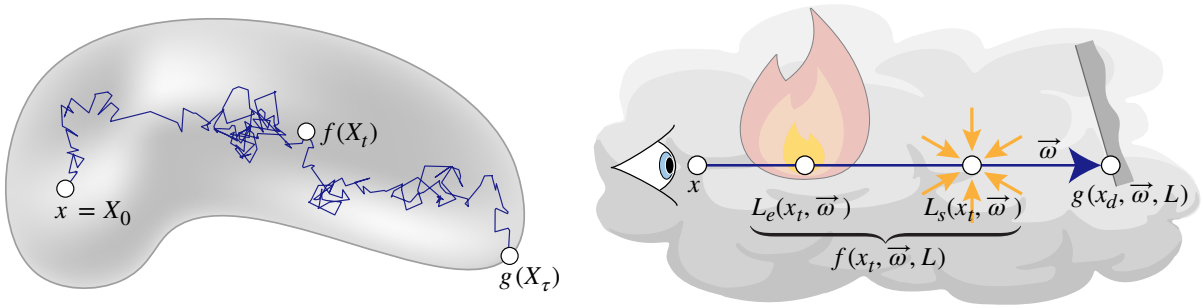


Figure 2.9: Left: The Feynman–Kac formula describes how a source term f and boundary data g contribute to the solution of an elliptic PDE with spatially varying coefficients along the trajectory of a random process X_t. Right: The volume rendering equation likewise describes the radiance $L(x, \vec{\omega})$ as a function of scattering, emission f, and the radiance g leaving the boundary, but along a straight light path rather than a random walk.

Similarity To The Volume Rendering Equation. In computer graphics, the radiative transport equation (RTE) [35] is used to describe the behavior of light in heterogeneous media that absorb, scatter and emit radiation (Fig. 2.9, *left*). Unlike Eq. 2.4, the RTE is only 1st order in space. It states that the *radiance* $L(x, \vec{\omega})$ at each point $x \in \Omega$ and in each direction $\vec{\omega} \in \mathbb{R}^n$ satisfies

$$\begin{aligned} \vec{\omega} \cdot \nabla L(x, \vec{\omega}) - \sigma(x) L(x, \vec{\omega}) &= -f(x, \vec{\omega}, L) \quad \text{on } \Omega, \\ L(x, \vec{\omega}) &= g(x, \vec{\omega}, L) \quad \text{on } \partial\Omega. \end{aligned} \tag{2.33}$$

This equation is recursive, since the source term $f(x, \vec{\omega}, L)$ depends on the radiance $L_s(x, \vec{\omega})$ *in-scattered* at x (as well as any *emission* $L_e(x, \vec{\omega})$); likewise, the function $g(x, \vec{\omega}, L)$ describes radiance leaving the boundary. The spatially varying *extinction coefficient* $\sigma(x)$ specifies the density of scattering or absorbing particles at x.

The integral representation of the RTE is called the *volume rendering equation (VRE)* [203, Ch. 15.1]. The VRE gives the radiance $L(x, \vec{\omega})$ as an integral along a ray $x_t := x - \vec{\omega} t$ of length d:

$$L(x, \vec{\omega}) = \mathrm{e}^{-\int_0^d \sigma(x_t)\,\mathrm{d}t} g(x_d, \vec{\omega}, L) \; + \; \int_0^d \mathrm{e}^{-\int_0^t \sigma(x_s)\,\mathrm{d}s} f(x_t, \vec{\omega}, L)\,\mathrm{d}t. \tag{2.34}$$

It shares a close resemblance with the Feynman–Kac formula (Eq. 2.31).

Delta tracking. The VRE is typically solved using *volumetric path tracing (VPT)* [141], but a spatially varying $\sigma(x)$ presents challenges akin to those for the Feynman–Kac formula: approximating the *transmittance function* $\exp(-\int_0^d \sigma(x_t)\,\mathrm{d}t)$ via explicit steps along x_t can yield significant error. *Delta tracking* [207, 270] instead rewrites Eq. 2.33 so that all spatial variation in the extinction coefficient $\sigma(x)$ is captured by a source term on the right-hand side—leaving only a constant absorption coefficient $\overline{\sigma} := \max(\sigma(x))$ [74, 139]:

$$\vec{\omega} \cdot \nabla L(x, \vec{\omega}) - \overline{\sigma} L(x, \vec{\omega}) = -(\underbrace{f(x, \vec{\omega}, L) + (\overline{\sigma} - \sigma(x)) L(x, \vec{\omega})}_{=:\, f'(x, \vec{\omega}, L)}). \tag{2.35}$$

Conceptually, fictitious *null matter* is added to the initially heterogeneous medium so that it has a constant density (Fig. 2.10, *left*). Eq. 2.35 then has the integral representation

$$L(x, \vec{\omega}) = \mathrm{e}^{-\overline{\sigma} d} g(x_d, \vec{\omega}, L) \; + \; \int_0^d \mathrm{e}^{-\overline{\sigma} t} f'(x_t, \vec{\omega}, L)\,\mathrm{d}t. \tag{2.36}$$

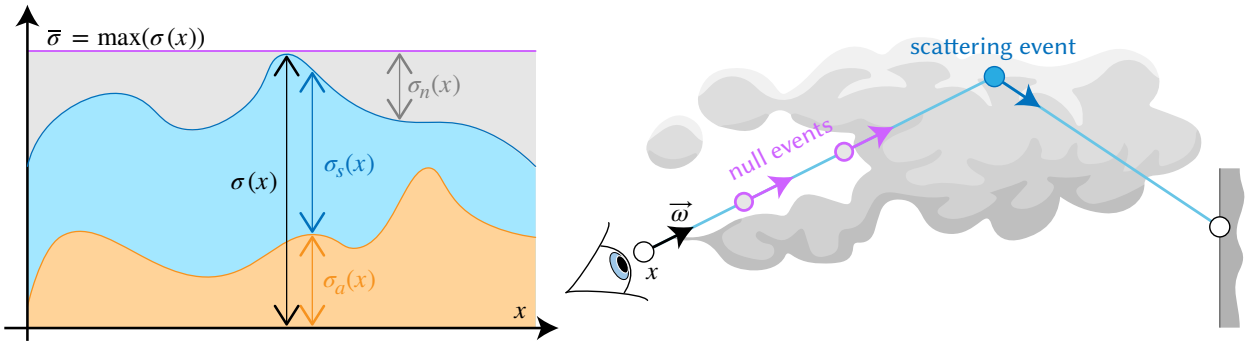


Figure 2.10: Left: The delta tracking method in volume rendering artificially fills a heterogeneous medium with fictitious null matter (indicated by $\sigma_n(x)$) so that the combined density $\overline{\sigma}$ is constant everywhere. Right: With VPT [70, Algorithm 2], we then re-weight the radiance by $\overline{\sigma} - \sigma(x)$ to account for the original heterogeneity in $\sigma(x)$, which corresponds to probabilistically sampling null-events inside the medium. Likewise in Sec. 5.2.1, our delta tracking WoS estimator will solve variable-coefficient PDEs by introducing null-events into the random walk.

This representation is more amenable to Monte Carlo integration, since the transmittance function $e^{-\overline{\sigma}t}$ can be evaluated in closed form. Spatial variations in $\sigma(x_t)$ are accounted for by weighting the radiance L by $\overline{\sigma} - \sigma(x_t)$ inside the modified source term f'.

In Ch. 5, we will derive a generalized mean value expression for the variable-coefficient PDEs from Sec. 2.1.1 by applying the delta tracking transformation to the Feynman–Kac formula. This will lead to an integral representation amenable to walk on spheres.

Chapter 3

Basic PDE Estimators

From here on, we use the Monte Carlo method to solve the PDEs from the previous chapter, via their integral formulations. In this chapter, we first explain how we can use Monte Carlo to numerically evaluate integrals that do not have analytic solutions. We then introduce Muller [174]'s walk on spheres algorithm for simulating Brownian motion (Sec. 2.3.1) in bounded domains, which allows us to apply Monte Carlo integration to mean value expressions (Sec. 2.2) for constant-coefficient PDEs with Dirichlet boundary conditions.

3.1 Background: Monte Carlo Integration

We review elementary facts about Monte Carlo integration, and refer to Veach [251, Chapter 2] and Pharr et al. [203, Chapter 13] for a more thorough introduction. The basic idea is that an integral can be estimated by simply sampling the integrand at randomly-chosen points, and averaging results from several trials. This makes Monte Carlo easy to implement and generally applicable to a wide variety of integrands, including those containing discontinuities or singularities. It is also the only practical numerical integration technique for high-dimensional integrals, since the performance of quadrature schemes becomes exponentially worse as dimensionality increases [251, Sec. 2.2]. As a randomized algorithm, Monte Carlo gives different results depending on the random numbers used, but on average the results are statistically close to the true answer. In particular, let ϕ be an L^1-integrable function on a domain A. Then the integral

$$I := \int_{\mathrm{A}} \phi(x)\,\mathrm{d}x \tag{3.1}$$

can be approximated by the *Monte Carlo estimator*

$$\widehat{I}_N \;:=\; |\mathrm{A}|\frac{1}{N}\sum_{i=1}^{N}\phi(x^i), \quad x^i \sim \mathscr{U}(\mathrm{A}), \tag{3.2}$$

where N is any positive integer, $x^i \sim \mathscr{U}(\mathrm{A})$ indicates that x^i are independent random samples drawn from the uniform distribution on A, and $|\mathrm{A}|$ denotes A's volume. In this text, we will express all PDE estimators as *single-sample estimators* $\widehat{I}$ (dropping the subscript $N = 1$ for brevity), with the expectation that their values will be averaged over many trials to improve accuracy.

Importantly, although $\widehat{I}_N$ is called an "estimator", it does not provide merely an estimate—rather, a well-designed estimator will give the *exact* value of the integral, in expectation. More precisely, an estimator is *unbiased* if its expected value equals the true value, $\mathbb{E}[\widehat{I}_N] = I$, for *any* number of samples N. We quantify the accuracy of an estimator using its expected squared error $\mathbb{E}[(\widehat{I}_N - I)^2]$, which for an unbiased estimator equals its *variance*

$$\mathrm{V}[\widehat{I}_N] := \mathbb{E}[(\widehat{I}_N - \mathbb{E}[\widehat{I}_N])^2]. \tag{3.3}$$

Assuming independent samples x^i, variance goes to zero at a rate of $O(1/N)$, irrespective of the dimensionality of the integral. This means that Monte Carlo algorithms converge at the rate of

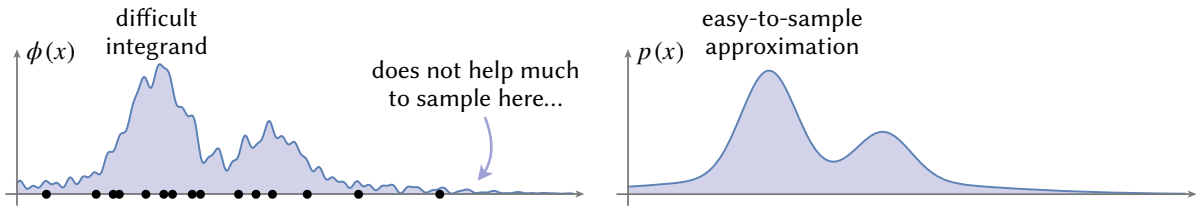


Figure 3.1: An importance sampling estimator concentrates samples where the integrand ϕ is large (left), often by using an easy-to-sample approximation of ϕ as its PDF p (right).

$O(1/\sqrt{N})$ to the correct result, *i.e.*, to cut error in half, we require four times as many samples. We must therefore apply variance reduction techniques, discussed next, to reduce noise in the estimated results given a fixed sample budget. In practice, we use *Welford's online algorithm* [267] to compute to a numerically stable estimate of the sample variance

$$\widehat{\mathrm{V}}[\widehat{I}_N] = \frac{1}{N-1}\sum_{i=1}^{N}\left[x^i - \left(\frac{1}{N}\sum_{j=1}^{N} x^j\right)\right]^2, \tag{3.4}$$

which is an unbiased estimator of Eq. 3.3. We say that two Monte Carlo estimators are *correlated* if they use the same set of independent samples to compute their respective sums.

3.1.1 Variance Reduction

Designing efficient estimators is of central importance to the adoption of Monte Carlo algorithms in downstream applications. The *efficiency* of an estimator $\mathrm{E}[\widehat{I}_N]$ is determined not just by its variance $\mathrm{V}[\widehat{I}_N]$, but also by the time $\mathrm{T}[\widehat{I}_N]$ needed to compute its value. Together, V and T define

$$\mathrm{E}[\widehat{I}_N] := \frac{1}{\mathrm{V}[\widehat{I}_N]\,\mathrm{T}[\widehat{I}_N]}, \tag{3.5}$$

which states that one estimator is more efficient than another if it either takes less time to produce the same variance, or if it produces less variance in the same amount of time.

Here we review variance reduction strategies from the broader Monte Carlo literature, which we will adapt to our setting in the next few chapters (but primarily in Ch. 7). This list is not exhaustive, as there likely exist other strategies for improving efficiency we do not consider here.

Importance Sampling

Let p^{A} be any PDF on the domain A that is nonzero on the support of the function ϕ. Then the integral of ϕ in Eq. 3.1 also equals the expected value of the estimator [203, Sec. 13.2]

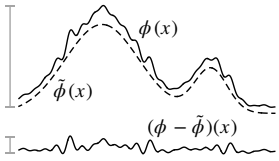
$$\widehat{I}_N^{\mathrm{IS}} := \frac{1}{N}\sum_{i=1}^{N}\frac{\phi(x^i)}{p^{\mathrm{A}}(x^i)}, \quad x^i \sim p^{\mathrm{A}}. \tag{3.6}$$

Importance sampling refers to concentrating samples in regions where ϕ has large values, by sampling from a density p^{A} that is similar to the integrand ϕ (Fig. 3.1). The theoretical best choice is to set $p^{\mathrm{A}} = c\phi$, where the constant of proportionality c equals 1 over the *unknown* value I of the integral $\int_{\mathrm{A}}\phi(x)\,\mathrm{d}x$ we are estimating. In this case, the expected squared error is zero, since the importance sampling estimator $\widehat{I}_N^{\mathrm{IS}}$ equals I. In practice, variance is reduced even when p^{A} only approximately matches ϕ, with importance sampling becoming especially helpful if integrands are localized or have singularities. In the remainder of this text, we will use Eq. 3.6 as the default estimator for all the PDEs we consider, and drop the superscript on $\widehat{I}^{\mathrm{IS}}$ for brevity.

Control Variates

Suppose we want to estimate the integral $\int_{\mathrm{A}} \phi(x)\,\mathrm{d}x$, and have a function $\tilde{\phi}(x)$ with known integral c. A *control variate* strategy then is

$$\widehat{I}_N^{\mathrm{CV}} := c + \frac{1}{N}\sum_{i=1}^{N} \frac{\phi(x^i) - \tilde{\phi}(x^i)}{p^{\mathrm{A}}(x^i)}, \quad x^i \sim p^{\mathrm{A}} \tag{3.7}$$



i.e., estimate the *difference* between ϕ and $\tilde{\phi}$, and shift by c. Intuitively, if $\tilde{\phi}$ is similar to ϕ, then the difference is a nearly constant function that will have smaller variance than ϕ itself (inset). This strategy will be useful for reducing noise when estimating derivatives of PDEs (Sec. 7.2).

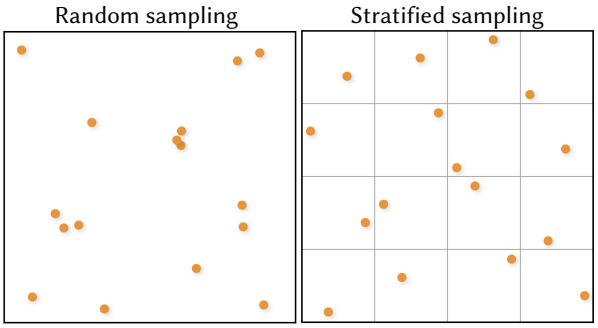
Antithetic Variates

Antithetic variates are most useful when the integrand ϕ is smooth and approximately linear. The basic idea is to add two estimators that use the same random samples and are negatively correlated, as the variance of the estimators together is smaller than if the samples are independent. Assume for simplicity that we want to estimate $\int_0^1 \phi(x)\,\mathrm{d}x$. Then, an *antithetic variate* strategy is

$$\widehat{I}_N^{\mathrm{AV}} := \frac{1}{N}\sum_{i=1}^{N} \frac{\phi(u^i) + \phi(1-u^i)}{2}, \quad x^i \sim \mathscr{U}([0,1]). \tag{3.8}$$

This estimator has zero variance when ϕ is a linear function, and extends easily to unit hypercubes $[0,1]^s$ of dimension s using sample pairs $\{u_1^i, \ldots, u_s^i\}$ and $\{1-u_1^i, \ldots, 1-u_s^i\}$. It can also be used on more general domains via a transformation between distributions [203, Sec. 13.5]. We will apply antithetic variates in conjunction with control variates to estimate PDE derivatives.

Stratified Sampling



Variance can also be reduced by subdividing the integration domain A into n nonoverlapping subdomains $\mathrm{a}_1, \mathrm{a}_2, \ldots, \mathrm{a}_n$, such that $\cup_{i=1}^n \mathrm{a}_i = \mathrm{A}$. Samples are then generated in each subdomain a_i using probability densities p^{a_i}. Compared to randomly sampling over all of A, this sort of stratified placement ensures that samples are more evenly distributed and do not clump together as much (see inset), making them less likely to miss important features of the integrand. A number of sophisticated sample placement schemes have been developed [251, Sec. 13.8] such as Latin hypercube sampling, orthogonal array sampling, and quasi-Monte Carlo (QMC) methods. QMC generally demonstrates asymptotically faster rates of convergence than standard Monte Carlo, but often at the cost of increased computation. We will stratify samples in the interior and on the boundary of a PDE domain in Sec. 7.5.

Adaptive Sampling

While importance sampling concentrates samples in regions where the integrand is large, adaptive sampling instead places samples in regions where the estimator has most variation. This

information can be computed directly from the estimated variance $\widehat{\mathrm{V}}[\widehat{I}_N]$ in Eq. 3.4 to inform placement of future samples. One can then change the sampling PDF as more information about the integrand is gathered while taking more samples. Unlike importance sampling, the main disadvantage of adaptive sampling is that it can produce biased results. However, bias can be justified if cost is significantly reduced—in Sec. 7.4, we will demonstrate that this is the case in regions where the integrand is smooth.

Sample Reuse And Caching

Generating samples from a probability density p^{A} can be expensive. For instance, when evaluating high-dimensional integrals with random walks (as we will see in Sec. 3.2), each sample depends on a sequence of nested decisions constituting k steps of a walk. In situations where several similar integrals need to be evaluated at nearby points in a domain, one can improve efficiency by reusing the same set of independent samples to evaluate more than one integral. Sample reuse introduces correlations between results which slow down the rate of convergence, but it can also greatly amortize the cost of generating expensive samples corresponding to, *e.g.*, long walks. Reuse may require storing samples in a cache, before their contribution can be shared between different estimators. We will develop a caching strategy for PDEs in Sec. 7.5.

Russian Roulette And Splitting

In particle transport [26, 102], *Russian roulette* and *splitting* are closely related techniques that improve the efficiency of an estimator by controlling the sample density in a domain. Unlike adaptive sampling, they do not introduce any bias, and ensure that each sample makes a significant contribution to the result.

Russian Roulette. Russian roulette provides an unbiased mechanism to discard samples that are expensive to evaluate but make little contribution to the estimator, *e.g.*, a random walk in a highly absorbing medium (Sec. 2.3.2). Russian roulette skips these samples by replacing the single-sample estimator $\widehat{I}$ with a new estimator of the form

$$\widehat{I}^{\,\mathrm{RR}} := \begin{cases} \widehat{I}/q & \text{with probability } q, \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

Here the survival probability q can be determined in any number of ways. For instance, q for a killed diffusion process (Sec. 2.3.1) can be based on the value of the exponential $\mathrm{e}^{-\sigma\tau}$ in Eq. 2.30. The estimator $\widehat{I}^{\,\mathrm{RR}}$ then has the same expected value as $\widehat{I}$:

$$\mathbb{E}[\widehat{I}^{\,\mathrm{RR}}] = q \cdot \frac{1}{q}\mathbb{E}[\widehat{I}] \;+\; (1-q)\cdot 0 = \mathbb{E}[\widehat{I}], \tag{3.10}$$

implying that it is free from bias whenever $\widehat{I}$ is unbiased. Though Russian roulette does not reduce variance, it can improve efficiency by reducing the average time spent evaluating $\widehat{I}$.

Splitting. Splitting on the other hand places more samples in important or high-contribution regions of the domain. In the context of random walks, this amounts to splitting a single walk into M new independent walks, each with $1/M$'th the contribution of the original walk. Concretely, the splitting estimator replaces $\widehat{I}$ with

$$\widehat{I}^{\,\mathrm{split}} := \frac{1}{M}\sum_{i=1}^{M} \widehat{I}_i, \tag{3.11}$$

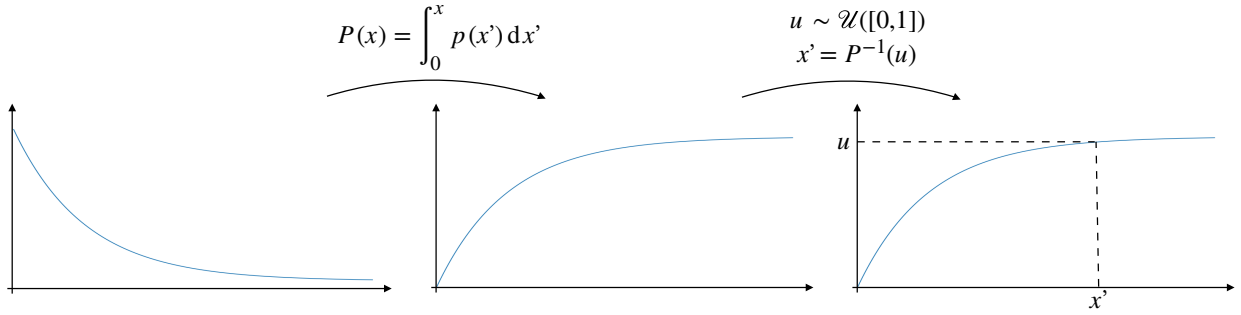


Figure 3.2: To generate samples from a probability density p, inverse transform sampling requires computing and inverting the cumulative density function P.

where the M estimates $\widehat{I}_i$ are independent from $\widehat{I}$ but have the same expectation. Like Russian roulette, this decomposition leads to an unbiased estimator $\widehat{I}^{\,\text{split}}$ via linearity of expectations. Since splitting promotes better exploration of the domain, variance is reduced but at the expense of increased runtime.

To improve efficiency, we will use Russian roulette to terminate random walks in domains with absorbing media (Sec. 3.2.2) and Robin boundary conditions (Sec. 4.3.4). In Sec. 7.7, we will also combine Russian roulette and splitting into a unified *weight window* strategy for PDE estimators with variable coefficients.

3.1.2 Sample Generation

To evaluate any of the Monte Carlo estimators from the previous section, we need to be able to draw random samples from a probability density. Here we describe a few standard sampling procedures which we will use later to sample from Green's functions. In the next section, we then turn to walk on spheres, which at its core is a sample generation algorithm as well, but specifically for generating realizations of Brownian motion in bounded domains.

Inverse Transform Sampling

Inverse transform sampling uses the following recipe to sample from a real-valued PDF p:

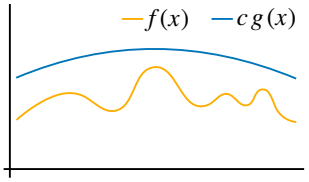
1. Compute the cumulative density function $P(x) = \int_0^x p(x')\,\mathrm{d}x'$.
2. Compute the inverse CDF $P^{-1}(y)$.
3. Generate a uniform random number $u \sim \mathscr{U}([0,1])$.
4. Generate a random sample $x' = P^{-1}(u)$.

As shown in Fig. 3.2 (*right*), the intuition behind this technique is that a value in $[0,1]$ on the vertical axis maps uniquely to the probability of an "outcome" x' on the horizontal axis—outcomes chosen for a set of uniform random numbers will therefore be distributed according to the PDF p. As an example, consider the PDF $p(x) = \sigma \mathrm{e}^{-\sigma x}$ of a killed diffusion process from Sec. 2.3.1, and its corresponding CDF $P(x) = 1 - \mathrm{e}^{-\sigma x}$. This CDF is easy to invert, $P^{-1}(y) = -\ln(1-y)/\sigma$, and thus easy to generate samples from, $x' = \ln(1-u)/\sigma$ for $u \sim \mathscr{U}([0,1])$.

For discrete outcomes, we can compute a CDF by summing up the known probability of each outcome, and using steps 3 and 4 to generate samples. The main challenge with using inverse

transform sampling on continuous PDFs is that it is not always possible to compute and invert CDFs. Rejection and weighted importance sampling relax these requirements.

Rejection Sampling



Rejection sampling draws samples from non-negative functions $f(x)$ that need not be normalized, *i.e.*, f does not have to integrate to 1, and hence is not necessarily a PDF. Assume we have access to a PDF g that is easy to sample from (via, *e.g.*, inverse transform sampling), and a constant c such that $f(x) \leq cg(x)$. We then repeat the following procedure till it returns a correctly distributed sample x':

1. Generate a uniform random number $u \sim \mathscr{U}([0,1])$.
2. Generate a sample $x' \sim g(x)$.
3. Return x' if $u < f(x')/cg(x')$.

Intuitively, when the PDF g is uniform, more samples are rejected in regions where f is small. Tighter bounds $cg(x)$ can therefore significantly improve performance.

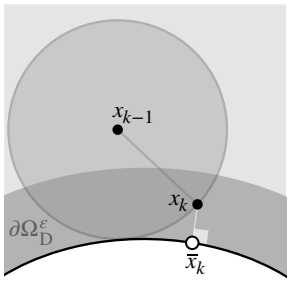
Weighted Importance Resampling

Assume we want to again sample according to an unnormalized function $f(x)$, but only have access to a stream of samples $\{x^1, x^2, \ldots\}$ generated using a PDF $g(x)$. We can choose a representative sample distributed proportionally to f by processing the stream one element at a time, and selecting—from the M samples seen so far—an x^i with probability $w(x^i)/\sum_{j=1}^{M} w(x^j)$, where the weight $w(x) := f(x)/g(x)$ [23, Alg. 3]. The next stream sample x^{M+1} replaces x^i with probability $w(x^{M+1})/\sum_{j=1}^{M+1} w(x^j)$. The stream length M need not be known ahead of time.

3.2 Walk On Spheres

Suppose we want to evaluate the solution to a basic Laplace equation $\Delta u = 0$ with Dirichlet boundary conditions g (Eq. 2.24) at some point $x_0 \in \Omega$. The mean value property (Eq. 2.13) says that $u(x_0)$ is equal to the average of u over the boundary of any ball $\mathrm{B}(x_0, R) \subset \Omega$. Alternatively, Kakutani's principle (Eq. 2.25) says that $u(x_0)$ equals the expected value of u where continuous Brownian random walks first hit the ball boundary $\partial\mathrm{B}$. We therefore have

$$u(x_0) = \frac{1}{|\partial \mathrm{B}(x_0, R)|} \int_{\partial \mathrm{B}(x_0, R)} u(z)\, \mathrm{d}z = \mathbb{E}[u(W_\tau)], \tag{3.12}$$



where by symmetry a random walk W_t starting at a point x_0 is equally likely to exit through any point on $\partial\mathrm{B}$—independent of how long the walk takes, or where it goes inside B (Fig. 3.3).

The mean value and stochastic perspectives both point to the same strategy for estimating $u(x_0)$: uniformly sample a point x_1 on a sphere around x_0. If x_1 is extremely close to the domain boundary (*i.e.*, within the ε-shell $\partial\Omega_{\mathrm{D}}^{\varepsilon}$), grab the boundary value $g(\bar{x}_1)$ at the closest point $\bar{x}_1 \in \partial\Omega_{\mathrm{D}}$ (inset). Otherwise, evaluate $u(x_1)$. Repeated evaluation of Eq. 3.12 results in a random walk on the points $x_0 \to x_1 \to \ldots$, where

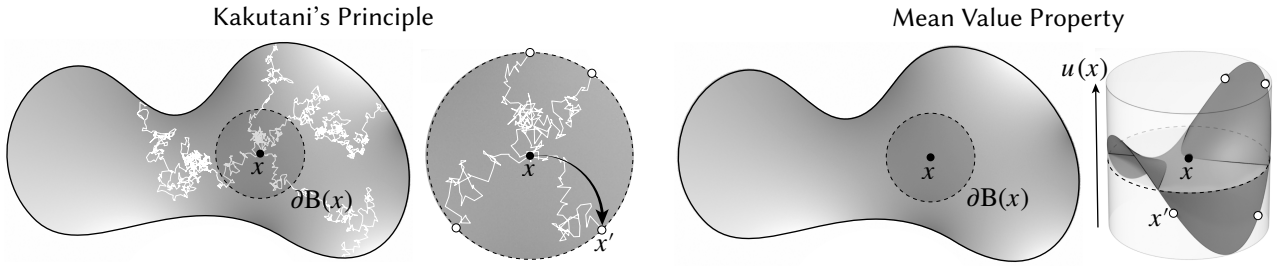


Figure 3.3: At any point x, the solution u to a Laplace problem is equal to the average boundary value reached by random walks (left), and the average value of u over a sphere $\partial\mathrm{B}(x)$ around x (right). Both quantities can be estimated by recursively sampling points x' on a sphere.

for $k \geq 0$, each point x_{k+1} lies on a sphere centered at the previous point x_k. This reasoning leads to the *recursive* WoS estimator by Muller [174]:

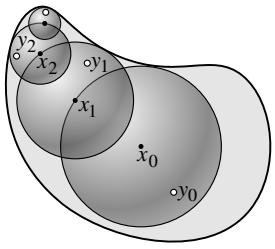
$$\widehat{u}(x_k) := \begin{cases} g(\overline{x}_k) & x_k \in \partial\Omega_{\mathrm{D}}^{\varepsilon}, \\ \frac{1}{|\partial\mathrm{B}(x_k,R)|} \frac{\widehat{u}(x_{k+1})}{p^{\partial\mathrm{B}}(x_{k+1})} & \text{otherwise.} \end{cases} \tag{3.13}$$

To achieve fast convergence to $\partial\Omega_{\mathrm{D}}$, we draw a single point x_{k+1} from the uniform density $p^{\partial\mathrm{B}} := 1/|\partial\mathrm{B}|$ on the *largest* sphere around x_k with radius $R = \|x_k - \overline{x}_k\|$ (Fig. 1.6), which can be computed efficiently via a closest point query (Sec. 6.1). In this case, one can show that the number of steps needed to reach the ε-shell $\partial\Omega_{\mathrm{D}}^{\varepsilon}$ is typically $O(\log 1/\varepsilon)$ [22], and that error vanishes quickly as ε is decreased (Sec. 6.3). Alg. 1 provides pseudocode.

Unlike SDE integration schemes (Sec. 2.3.1), WoS provides an exact statistical simulation of Brownian motion without introducing discretization error at every step of a random walk. Moreover, numerical quadrature is not a practical alternative to Monte Carlo here, since the recursively expanded integration domain is high-dimensional. In the remainder of this section, we describe known extensions of WoS to other constant coefficient PDEs such as the Poisson and screened Poisson equations. We also discuss estimation of derivatives, and treatment of reflecting boundary conditions.

3.2.1 Poisson Equation

We can solve a Poisson equation (Eq. 2.2) by incorporating a source term $f : \Omega \to \mathbb{R}$ into the WoS estimator as follows:



$$\widehat{u}(x_k) := \begin{cases} g(\overline{x}_k) & x_k \in \partial\Omega_{\mathrm{D}}^{\varepsilon}, \\ \frac{1}{|\partial\mathrm{B}(x_k,R)|} \frac{\widehat{u}(x_{k+1})}{p^{\partial\mathrm{B}}(x_{k+1})} + \frac{G^{\mathrm{B}(x_k,R)}(x_k,\, y_{k+1}) f(y_{k+1})}{p^{\mathrm{B}}(y_{k+1})} & \text{otherwise.} \end{cases} \tag{3.14}$$

A good default strategy for the second term, shown in the inset, is to importance sample the Green's function G^{B} at every step of a walk by drawing a single sample $y_{k+1} \in \mathrm{B}(x_k, R)$ from the density $p^{\mathrm{B}} := G^{\mathrm{B}}(x_k, y_{k+1})/|G^{\mathrm{B}}(x_k)|$. Here $|G^{\mathrm{B}}(x_k)|$ is the integral of G^{B} over B (App. A.6). We can use either rejection sampling or inverse transform sampling to generate samples from G^{B} (App. A.1.2), and can use more than one sample at each step to reduce variance if needed. We refer to Delaurentis and Romero [51, Sec. 2] for a proof that this single-sample estimator converges to the true solution $u(x)$.

ALGORITHM 1: WalkOnSpheres(x, ε)

Input: Starting position $x \in \Omega$ of random walk, and ε-shell.
Output: Single-sample MC estimate $\widehat{u}(x)$ of Poisson equation with Dirichlet conditions.

$d, \overline{x} \leftarrow$ ClosestPt$(\partial\Omega_{\mathrm{D}}, x)$ ▷*Compute distance to absorbing boundary $\partial\Omega_{\mathrm{D}}$ (Sec. 6.1)*
if $d < \varepsilon$ **then return** $g(\overline{x})$ ▷*Return boundary value g at closest pt $\overline{x}$ if $x \in \partial\Omega_{\mathrm{D}}^{\varepsilon}$*
$v \leftarrow$ SampleUnitSphere() ▷*Sample direction v uniformly on unit sphere*
$p \leftarrow x + d\, v$ ▷*Set next walk position on sphere with radius d*
$\widehat{I}_f \leftarrow$ SourceEstimate(x, d) ▷*Estimate contribution from source term f (Sec. 3.2.1)*
return WalkOnSpheres$(p, \varepsilon) + \widehat{I}_f$ ▷*Repeat from next walk position p*

3.2.2 Screened Poisson Equation

To incorporate a constant absorption coefficient $\sigma \in \mathbb{R}_{\geq 0}$ (Eq. 2.3), we simply replace the functions $1/|\partial \mathrm{B}|$ and G^{B} in the estimator for a Poisson equation with the corresponding Poisson kernel $P^{\sigma,\mathrm{B}}$ and Green's function $G^{\sigma,\mathrm{B}}$ for a screened Poisson equation [61]; App. A.2.2 provides explicit expressions. As discussed in Sec. 2.3.2, a nonzero σ dampens the PDE solution, which can also be observed through the following relation (from Eq. A.17) between Poisson kernels:

$$P^{\sigma,\mathrm{B}}(x, y) = \frac{1}{|\partial \mathrm{B}(x, R)|} Q^{\sigma,\mathrm{B}}(x, y), \tag{3.15}$$

where $Q^{\sigma,\mathrm{B}} \in [0, 1)$ for $\sigma > 0$. Hence, if we use the uniform density $p^{\mathrm{B}} = 1/|\partial \mathrm{B}|$ to sample the next walk position x_{k+1}, then the solution estimate $\widehat{u}$ accumulates a *throughput* $\prod_k Q^{\sigma,\mathrm{B}}(x_k, x_{k+1})$ over k steps that downweights both the boundary and source contribution g and f, respectively.

We use the term throughput in analogy with the throughput of a light path in Monte Carlo rendering [203]—in the context of a screened Poisson equation, the throughput of a walk equals the probability with which a killed Brownian motion (Sec. 2.3.1) does not get absorbed in the domain Ω. With WoS, we can use Russian roulette (Sec. 3.1.1) to realize absorptions: we terminate walks at step k with probability $1 - Q^{\sigma,\mathrm{B}}(x_k, x_{k+1})$, and cancel out the contribution $Q^{\sigma,\mathrm{B}}(x_k, x_{k+1})$ in the walks that survive. This allows us to terminate walks early, instead of waiting for them to reach $\partial\Omega_{\mathrm{D}}^{\varepsilon}$ while their throughput continues to shrink. In practice, Russian roulette provides large efficiency gains when solving screened Poisson equations with WoS.

3.2.3 Biharmonic Equation

Several algorithms in geometric computing require solving two or more PDEs that depend on one another, *e.g.*, computing deformations [240] and geodesic distances [45], solving eigenvalue problems via power iterations. Here we consider the simpler example of a biharmonic equation

$$\begin{aligned} \Delta^2 u &= 0 \quad \text{on } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \\ \Delta u &= h \quad \text{on } \partial\Omega, \end{aligned} \tag{3.16}$$

which can be formulated as a system of two 2nd order PDEs via the substitution $v := \Delta u$:

$$\begin{aligned} \Delta u &= v \quad \text{on } \Omega, \qquad & \Delta v &= 0 \quad \text{on } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \qquad & v &= h \quad \text{on } \partial\Omega. \end{aligned} \tag{3.17}$$

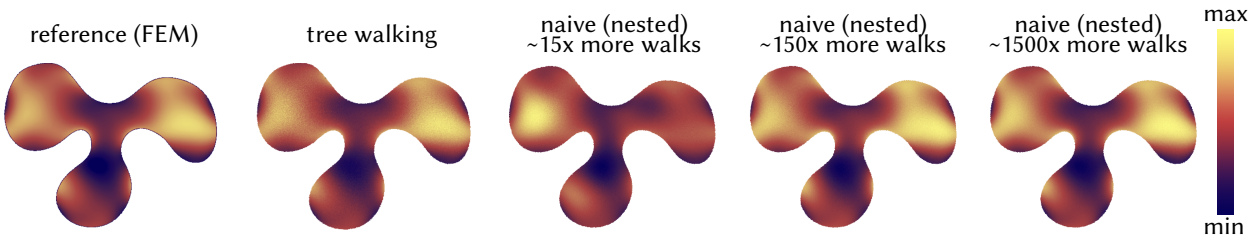
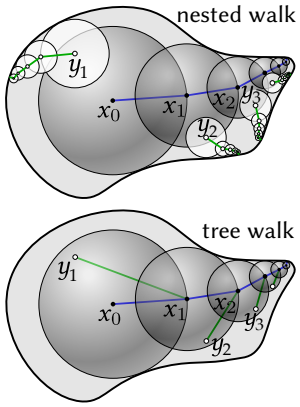


Figure 3.4: A comparison of the "tree walking" strategy for a biharmonic equation to a naïve nested strategy with the same number of outer walks, and 1, 10, and 100 inner walks; for this example each outer walk takes about 15 steps. The tree walking result is slightly noisier, but many times more efficient.



To evaluate u at a point $x_0 \in \Omega$, we can apply the standard Poisson estimator (Eq. 3.14), using the Laplace estimator (Eq. 3.13) to estimate v at each step of a walk. However, this naïve strategy is quite expensive since we now have to simulate whole walks inside each step (inset, *top*). This requires $O(SN^2)$ steps, where N is the number of walks used for the Poisson and Laplace estimators, and each walk takes $O(S)$ steps.

To address the inefficiency above, we propose a "tree walking" strategy that re-uses partial walks to reduce cost to $O(SN)$ [221, Sec. 4.3]. In particular, at each step x_k of the "outer" walk, we still use a single point y_{k+1} to sample the source term. But rather than using the basic WoS estimator for $v(y_{k+1})$ (Eq. 3.13), we instead use the estimate $\widehat{v}(y_{k+1}) = P^{\mathrm{B}}(y_{k+1}, x_{k+1})h(\overline{x}_N)$, where P^{B} is the *off-centered* Poisson kernel for the ball B (Sec. 2.2.1), and x_N is the final point in the walk used to estimate $u(x_0)$. In other words, we connect a walk of length one (from y_{k+1} to x_{k+1}) to the longer walk from x_{k+1} to x_N, then use the boundary value (inset, *bottom*). Though the walks are now more strongly correlated due to reuse (x_{k+1} is connected to both x_k and y_{k+1}), they are also significantly cheaper to compute—in practice, we get reduced variance for equal compute time (Fig. 3.4). Similar strategies can in principle be applied to other nested sequences of PDEs.

3.2.4 Exterior Problems

The WoS estimators from the previous sections can be used to solve PDEs on the domain *exterior*, *i.e.*, the complement of Ω in $\mathbb{R}^N$. Unfortunately, there is a non-zero probability for a random walk to wander off to infinity in an open domain, or in the exterior of a closed domain; this corresponds to the *non-recurrent* behavior of Brownian motion in dimensions greater than 1 [27, Ch. 2]. A simple strategy then is to apply Russian roulette to terminate walks that wander far from the domain boundary—examples are shown in Fig. 1.7. Nabizadeh et al. [179] instead propose an approach based on the spherical inversion of the domain via a *Kelvin transformation*, and demonstrate that WoS solves exterior problems with significantly lower variance under this inversion. We refer to their work for further details.

3.2.5 Spatial Derivatives

One often requires computing not just the solution to a PDE, but also its spatial derivatives for tasks involving, *e.g.*, shape optimization [204] and gradient flows (Fig. 8.5 & 7.10). Surprisingly little has been said about estimating derivatives via WoS—one such work is by Elepov and Mikhailov [61], which briefly discusses gradients but ignores other differential operators,

higher-order derivatives, and variance reduction strategies (Sec. 7.2 & 7.5). We provide basic estimators for spatial derivatives [221, Sec. 3] derived using harmonic analysis [9]. The more general framework of *Malliavin calculus* [17] provides tools for also computing sensitivities with respect to quantities such as PDE coefficients (termed *Greeks* in financial engineering [71]).

Gradient

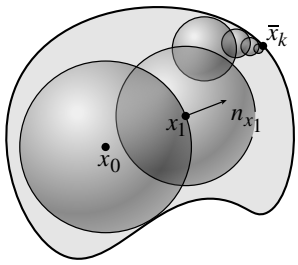
We can express the gradient of a harmonic function u, with respect to an evaluation point $x \in \Omega$, via a mean value-like integral [221, App. A]

$$\nabla_x u(x) = \frac{1}{|\mathrm{B}(x,R)|} \int_{\partial \mathrm{B}(x,R)} u(y)\; n_y \,\mathrm{d}y, \tag{3.18}$$

where for a d-dimensional ball of radius R, $n_y \coloneqq (y-x)/R$ is the outward unit normal at y. A basic WoS estimator for the gradient then is

$$\widehat{\nabla_{x_0} u}(x_0) := \frac{d}{R} \widehat{u}(x_1)\; n_{x_1}, \quad x_1 \sim \mathscr{U}(\partial B(x_0, R)), \tag{3.19}$$

i.e., just uniformly sample a point x_1 on a ball around x_0, and multiply the normal n_{x_1} by the usual WoS estimate for $u(x_1)$ (Eq. 3.13). The coefficient d/R comes from the surface area to volume ratio for an d-dimensional ball. We note that estimating the gradient adds virtually no cost on top of computing the solution at x_0, apart from a multiplication with the normal. Eq. 3.19 also makes no assumptions about the boundary conditions imposed on the domain boundary $\partial\Omega$—it simply requires access to an estimate of u at $x_1 \in \Omega$. However, Eq. 3.19 does not apply when $x_0 \in \partial\Omega_{\mathrm{D}}$, since $R = 0$.



More generally, the gradient of the (off-centered) mean value property for a screened Poisson equation (Sec. 2.2.1) is given by

$$\nabla_x u(x) = \int_{\partial \mathrm{B}(c,R)} \nabla_x P^{\sigma,\mathrm{B}}(x,z)\; u(z)\,\mathrm{d}z \;+\; \int_{\mathrm{B}(c,R)} \nabla_x G^{\sigma,\mathrm{B}}(x,y)\; f(y)\,\mathrm{d}y, \tag{3.20}$$

where c is the center of the ball B. The WoS estimator for this expression requires only a single sample for the second integral at the first step of a walk. We recover the spatial gradient for a Poisson equation when $\sigma = 0$, and the gradient for the Laplace equation in Eq. 3.18 when $f = 0$ and $c = x$. App. A gives expressions for the functions $\nabla_x P$ and $\nabla_x G$, while Ch. 7 provides variance reduction strategies for gradient estimators.

First-Order Differential Operators

Given an estimate for the gradient $\nabla_x u$, an estimate of the *directional derivative* $D_Z u$ along any direction $Z \in \mathbb{R}^n$ is then given by

$$D_Z u(x) = Z \cdot \nabla_x u(x). \tag{3.21}$$

All other first-order differential operators, such as divergence or curl, can be expressed via the partial derivatives $\partial u/\partial e_i = D_{e_i} u$ along the coordinate directions $e_1, \ldots, e_n$.

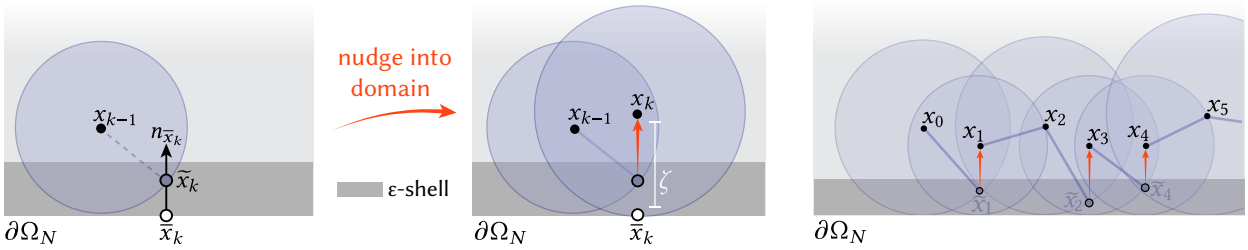


Figure 3.5: To simulate reflecting random walks with WoS, a standard approach [156, 160] is to offset a walk that approaches the Neumann boundary back into the domain by a fixed distance along the inward normal n to the boundary (left). This approach introduces discretization error into the reflecting walk simulation. Moreover, the resulting walks tend to stick to the boundary as they are attracted to it, leading to long walk lengths (right).

Higher Order Derivatives

If u is the solution to a Poisson equation (Eq. 2.2), then its Hessian $\mathbb{H}u$ can be expressed via the integral formula

$$\begin{aligned}\mathbb{H}u(x) \;=\; & \tfrac{d^2}{R^4}\tfrac{1}{|\partial \mathrm{B}(x,R)|}\int_{\partial \mathrm{B}(x,R)} u(y)(y-x)(y-x)^{\mathrm{T}}\,\mathrm{d}y \\ & - \tfrac{d}{R^2}\tfrac{1}{|\partial \mathrm{B}(x,R)|}\int_{\partial \mathrm{B}(x,R)} u(y)\;\mathbb{I}\,\mathrm{d}y \\ & + \int_{\mathrm{B}(x,R)} \mathbb{H}G(x,y) f(y)\,\mathrm{d}y,\end{aligned} \tag{3.22}$$

where $\mathbb{I} \in \mathbb{R}^{d\times d}$ is the identity matrix. A single-sample WoS estimator could in principle be written as

$$\widehat{u}(x_1)\left(\frac{d^2}{R^4}(x_1-x_0)(x_1-x_0)^{\mathrm{T}} - \frac{d}{R^2}\mathbb{I}\right) \;+\; |\mathrm{B}(x_0,R)|\mathbb{H}G(x_0,y_1)f(y_1), \tag{3.23}$$

where $x_1 \sim \mathscr{U}(\partial \mathrm{B}(x_0,R))$ and $y_1 \sim \mathscr{U}(\mathrm{B}(x_0,R))$. Here we run into some difficulty as the Hessian of the Green's function $\mathbb{H}G$ involves terms that behave like Dirac deltas: without an importance sampling strategy akin to those used for point sources (Sec. 7.1), important contributions will be missed. However, we can apply integration by parts to obtain a different expression for this term, namely

$$\int_{\mathrm{B}(x,R)} f(y)\big(\psi(x,y)(y-x)(y-x)^{\mathrm{T}} - \phi(x,y)\;\mathbb{I}\big) - \nabla_y f(y)\nabla_x G(x,y)\,\mathrm{d}y, \tag{3.24}$$

where ψ and ϕ are functions that depend on the particular choice of Green's function (see [221, App. B.2.2]), and ∇_y denotes the gradient with respect to y. This quantity can be estimated via a single Monte Carlo sample, though we now also need access to the gradient of the source function f. As with the gradient estimator (Sec. 3.2.5), other 2nd-order differential operators can be estimated via a Hessian estimate. See Fig. 7.4 for a numerical example and convergence plot.

3.2.6 Reflecting Boundary Conditions

So far, we have presented WoS estimators for PDEs with absorbing Dirichlet boundary conditions. For mixed boundary value problems containing reflecting Neumann conditions (Eq. 2.1), the standard approach [160] is again to terminate walks when they reach the ε-shell $\partial\Omega^{\varepsilon}_{\mathrm{D}}$. However, if a walk ever reaches a point $\widetilde{x}_k$ in the ε-shell $\partial\Omega^{\varepsilon}_{\mathrm{N}}$ around the Neumann boundary, then the Neumann data h at the closest point $\overline{x}_k \in \partial\Omega_{\mathrm{N}}$ is approximated via finite differences, *e.g.*,

$$h(\overline{x}_k) \approx \frac{u(\overline{x}_k + \zeta n_{\overline{x}_k}) - u(\overline{x}_k)}{\zeta}, \tag{3.25}$$

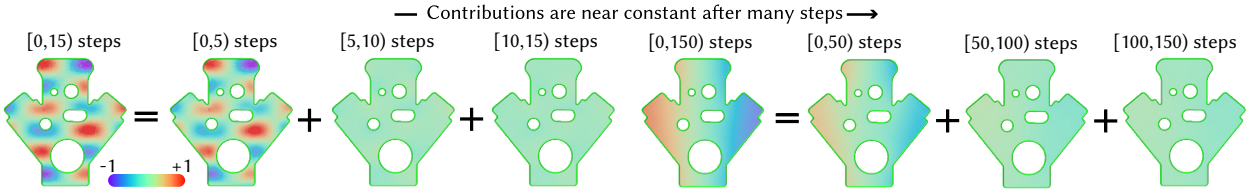


Figure 3.6: For a Poisson equation with pure Neumann conditions, local details in the PDE solution are often resolved by the first few steps of a Brownian random walk (simulated here using the walk on stars algorithm from Ch. 4), with near-constant contributions from later steps (left). However, more steps are typically needed to resolve lower frequency global details (right).

where $\zeta > \varepsilon$ is a constant. The solution estimate at x_k is then

$$\widehat{u}(x_k) := \widehat{u}(\overline{x}_k + \zeta n_{\overline{x}_k}) - \zeta h(\overline{x}_k) \approx u(\overline{x}_k). \tag{3.26}$$

In other words, $-\zeta h(\overline{x}_k)$ is added to the running estimate, and the walk continues as usual from the point $\overline{x}_k + \zeta n_{\overline{x}_k}$ obtained by nudging $\overline{x}_k$ back into the domain by a fixed distance ζ along the inward unit normal (Fig. 3.5, left). Mascagni and Simonov [160] call this procedure a *boundary reflection*; Maire and Tanré [156] and Zhou et al. [283] provide more sophisticated approximations using higher-order differences. Similar approximations are also available for Robin boundary conditions [236, 282].

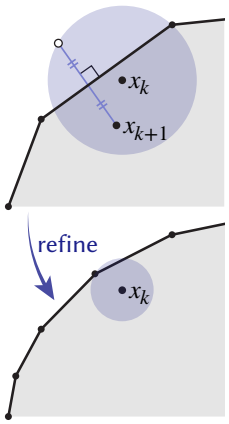
Unfortunately, such reflections are often impractical for problems with a large Neumann boundary $\partial\Omega_N$: the finite difference approximation introduces significant bias if ζ is much larger than ε. Yet if ζ is only slightly larger than ε, then random walks "stick" to $\partial\Omega_N^\varepsilon$, and take many small steps before escaping toward the interior (Fig. 3.5, right). Fig. 8.16 (*left*) shows that, in practice, boundary reflections yield both slow runtime and large accumulated bias. In the next chapter, we develop a walk on stars estimator for BVPs with mixed boundary conditions [167, 224]. WoSt avoids these issues by considering larger spheres that contain the Neumann and Robin boundaries, which greatly improve both accuracy and efficiency.

Pure Neumann Conditions The solution to a Poisson equation with pure Neumann boundary conditions is determined only up to an additive constant. From the random walk perspective, there is no Dirichlet boundary to terminate on, hence contributions from the Neumann data h accumulate forever. However, shorter walks tend to resolve high-frequency details in the solution, whereas the contribution from independent longer walks is more spatially uniform (see Fig. 3.6). Based on this observation, Maire and Tanré [156] describe a WoS estimator that stops the simulation once walks become longer than a certain length, implicitly pinning an additive constant to the solution. For WoSt we will instead apply *Tikhonov regularization*, which makes the solution unique by adding a small absorption coefficient σ to the PDE (resulting in a screened Poisson equation). In particular, we will switch to this PDE when a walk gets longer than a user-specified length (Fig. 7.5), which adds a small but controlled amount of bias. As described in Sec. 3.2.2, we can then terminate walks via Russian roulette, using a survival probability proportional to the Poisson kernel of a screened Poisson equation.

Chapter 4

Estimators For PDEs With Mixed Boundary Conditions

The original walk on spheres method solves Dirichlet problems by simulating random walks that ultimately get absorbed into the boundary. Rather than simulate many small steps of an isotropic Brownian motion (Fig. 2.7, *left*), this process is greatly accelerated by sampling the next point from the largest empty ball around the current walk position (Fig. 4.1, *left*). In this chapter, we take a basic but important step forward by developing practical strategies for incorporating Neumann and Robin boundary conditions (Eq. 2.1), which are an essential component of virtually every real physical model. To model these boundary conditions, one must additionally simulate *reflecting* random walks that "bounce" off the boundary (Fig. 2.7, *center* & *right*) [86, 87]. In a half space, reflecting walks amount to just taking the absolute value of Brownian motion in one coordinate direction. Hence, for polyhedral domains, a naïve strategy for simulating reflections is to sample the largest ball that intersects only a single boundary face, and to perform a reflection across the boundary plane if the sampled point falls outside the domain (see inset). However, the efficiency of this strategy drops quickly as the boundary mesh is refined.



Our strategy, which we call *walk on stars (WoSt)* [167, 224], is both more efficient and more general—it is a Monte Carlo estimator for the boundary integral equation of a Laplace problem (Sec. 2.2). In short, we identify a large *star-shaped region* around the current walk position, and sample a point on its boundary by picking a random direction (Fig. 4.1, *center* & *right*). Like WoS, WoSt takes large steps inside the domain to quickly reach the Dirichlet boundary (these techniques are in fact *equivalent* for pure Dirichlet problems). Yet unlike prior WoS schemes for Neumann and Robin boundaries (Sec. 3.2.6), WoSt can also take large steps that are independent of the level of tessellation near a reflecting boundary. In Sec. 8.3, we will demonstrate that this strategy works reliably in both convex and non-convex domains without incurring large bias or variance, unlike other grid-free Monte Carlo approaches [63, 217, 235, 236, 242].

To use WoSt, the only question that must be answered is: how do we find star-shaped regions? We propose one strategy here, using the *visibility silhouette*, which is easy to implement efficiently without much overhead (Sec. 6.2). Fundamentally, however, the WoSt approach relies only on the use of star-shaped regions, and not on any particular method used to compute them, or on any particular representation of the domain boundary. Importantly, WoSt requires only few changes to an existing WoS implementation, and provides the same advantages as WoS such as progressive and output sensitive evaluation, trivial parallelization, and robustness to defective geometry, while being applicable to a broader class of problems (see, *e.g.*, Fig. 1.9 & 4.2).

We first develop the WoSt estimator for Neumann problems in Sec. 4.2, and then describe modifications for Robin conditions in Sec. 4.3. We limit the discussion to Poisson equations here, and address more general linear elliptic PDEs in the next chapter. We also assume for simplicity that the domain Ω is a compact subset of $\mathbb{R}^N$, and provide extensions to open domains and double-sided boundary in App. B.

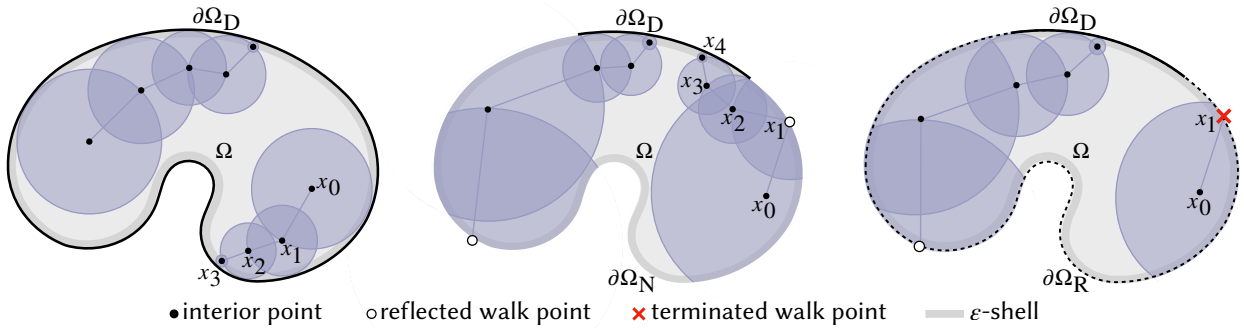
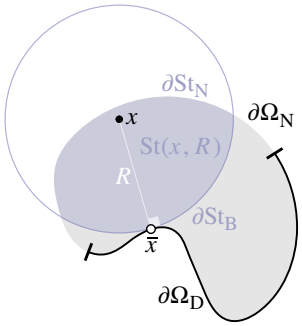


Figure 4.1: WoSt solves BVPs with any combination of Dirichlet, Neumann and Robin boundary conditions. Left: With a purely absorbing Dirichlet boundary $\partial\Omega_D$, WoSt is equivalent to WoS, which repeatedly jumps to a random point on the largest sphere around the current walk location, and terminates when the walk enters an ε-shell $\partial\Omega_D^\varepsilon$. Center: For a reflecting Neumann boundary $\partial\Omega_N$, WoSt generalizes to using star-shaped regions—specifically a sphere containing a subset of $\partial\Omega_N$. The next walk location is determined by intersecting a uniformly sampled ray direction against the current sphere and the visible portion of $\partial\Omega_N$ it contains, and picking the first hit point. Right: For a Robin boundary that is both reflecting and absorbing, the walk can additionally terminate on $\partial\Omega_R$.

4.1 Star-Shaped Regions

In lieu of balls, WoSt considers regions that are star-shaped with respect to a point $x \in \Omega$, *i.e.*, regions whose boundary is visible from x. Though in principle any star-shaped region could be used, we use regions $\mathrm{St}(x, R)$ given by the intersection $\mathrm{B}(x, R) \cap \Omega$ containing x, for a particular choice of radius R for the ball B (see Sec. 4.2.2). Similar to Eq. 2.1, we partition the region boundary into a Neumann part $\partial\mathrm{St}_\mathrm{N} := \partial\Omega_\mathrm{N} \cap \partial\mathrm{St}$ with prescribed normal derivatives $\partial u/\partial n = h$, and a spherical part $\partial\mathrm{St}_\mathrm{B} := \partial\mathrm{B} \cap \partial\mathrm{St}$. For a Robin boundary $\partial\Omega_\mathrm{R}$, we likewise define $\partial\mathrm{St}_\mathrm{R} := \partial\Omega_\mathrm{R} \cap \partial\mathrm{St}$ with prescribed values $\partial u/\partial n + \mu u = \ell$.



4.2 Walk On Stars With Neumann Conditions

Letting A := St and C := B in Eq. 2.10, the boundary integral for a Poisson equation becomes

$$\alpha(x)u(x) = \int_{\partial\mathrm{St}(x,R)} P^\mathrm{B}(x,z)u(z)\,\mathrm{d}z \;+\; \int_{\partial\mathrm{St}_\mathrm{N}(x,R)} G^\mathrm{B}(x,z)h(z)\,\mathrm{d}z \;+\; \int_{\mathrm{St}(x,R)} G^\mathrm{B}(x,y)f(y)\,\mathrm{d}y. \quad (4.1)$$

As with the mean value property used by WoS (Eq. 2.13), the solution value $u(z)$ is the only unknown in this equation: at points $z \in \partial\mathrm{St}_\mathrm{N}$, the normal derivative $\partial u/\partial n$ is given by the fixed Neumann data h along $\partial\Omega_\mathrm{N}$, and is not needed at points $z \in \partial\mathrm{St}_\mathrm{B}$ where $G^\mathrm{B}(x,z) = 0$. Since only one quantity is unknown, estimators for this equation need not branch.

Simonov [235], and later Ermakov and Sipin [63], take a parallel approach on domains Ω with *convex* Neumann boundaries $\partial\Omega_\mathrm{N}$. In particular, they use regions formed by intersecting Ω with a ball B whose radius is the distance to the Dirichlet boundary, $d_\mathrm{Dirichlet} := \|x - \overline{x}\|$. Hence, B can contain a subset of the Neumann boundary $\partial\Omega_\mathrm{N}$. In the convex case, such regions are automatically star-shaped. To handle arbitrary domains, WoSt instead uses visibility information to obtain star-shaped regions even near nonconvex Neumann boundaries (which in general can yield a radius $R \leq d_\mathrm{Dirichlet}$), as we will see in Sec. 4.2.2.

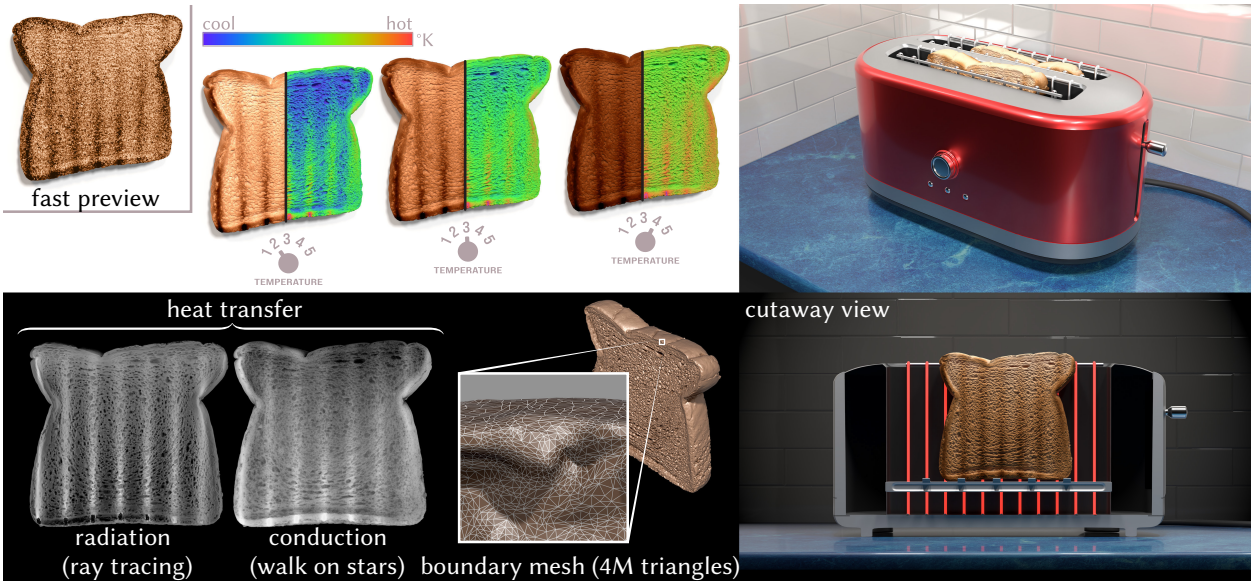


Figure 4.2: The walk on stars method handles mixed boundary conditions, enabling it to model a richer class of problems than the original walk on spheres method. Here for instance we simulate conductive heat transfer from a toaster with Dirichlet conditions to a piece of bread with Neumann conditions, by solving a Laplace equation (top and bottom right), complementing the radiative transfer computed via ray tracing (bottom left). As with ray tracing, we can simulate directly on the full high-resolution data (bottom center) without generating a volume mesh or forming a global stiffness matrix. Since results are progressive, we can get a preview of how the toast will look faster than it takes to toast a real piece of bread (top left).

4.2.1 Monte Carlo Estimation

The Walk On Stars Estimator With Dirichlet-Neumann Conditions

A single-sample Monte Carlo estimator for Eq. 4.1 is given by

$$\widehat{u}(x_k) = \frac{P^{\mathrm{B}}(x_k, x_{k+1})\ \widehat{u}(x_{k+1})}{\alpha(x_k)\ p^{\partial \mathrm{St}(x_k,R)}(x_{k+1})} + \frac{G^{\mathrm{B}}(x_k, z_{k+1})\ h(z_{k+1})}{\alpha(x_k)\ p^{\partial \mathrm{St}_{\mathrm{N}}(x_k,R)}(z_{k+1})} + \frac{G^{\mathrm{B}}(x_k, y_{k+1})\ f(y_{k+1})}{\alpha(x_k)\ p^{\mathrm{St}(x_k,R)}(y_{k+1})}, \tag{4.2}$$

where for $k \geq 0$,

- the points $x_{k+1} \in \partial\mathrm{St}$, $z_{k+1} \in \partial\mathrm{St}_{\mathrm{N}}$, and $y_{k+1} \in \mathrm{St}$ are sampled from the probability densities $p^{\partial\mathrm{St}}$, $p^{\partial\mathrm{St}_{\mathrm{N}}}$ and p^{St} (respectively).
- R is chosen so that $\mathrm{St}(x_k, R)$ is star-shaped.

The WoSt estimator is *recursive* as $\widehat{u}$ appears on both sides of Eq. 4.2. Applying it iteratively leads to a random walk from one star-shaped region to another—hence the name *walk on stars*. At a high level, each step of WoSt accumulates contributions from the (non-recursive) Neumann data h and source term f. For mixed Dirichlet-Neumann problems, the walk terminates when it enters the ε-shell $\partial\Omega^{\varepsilon}_{\mathrm{D}}$, using the Dirichlet data g at the closest point $\overline{x}_k \in \partial\Omega_{\mathrm{D}}$ as the solution estimate, *i.e.*, $\widehat{u}(x_k) \coloneqq g(\overline{x}_k)$. For pure Dirichlet problems, WoSt reduces to WoS; for pure Neumann problems, we apply Tikhonov regularization to terminate walks at the expense of some bias (Sec. 7.3). We first discuss how to sample the next step x_{k+1} and choose the radius R (Sec. 4.2.2), followed by sampling procedures for h and f (Sec. 4.2.3 & 4.2.4).

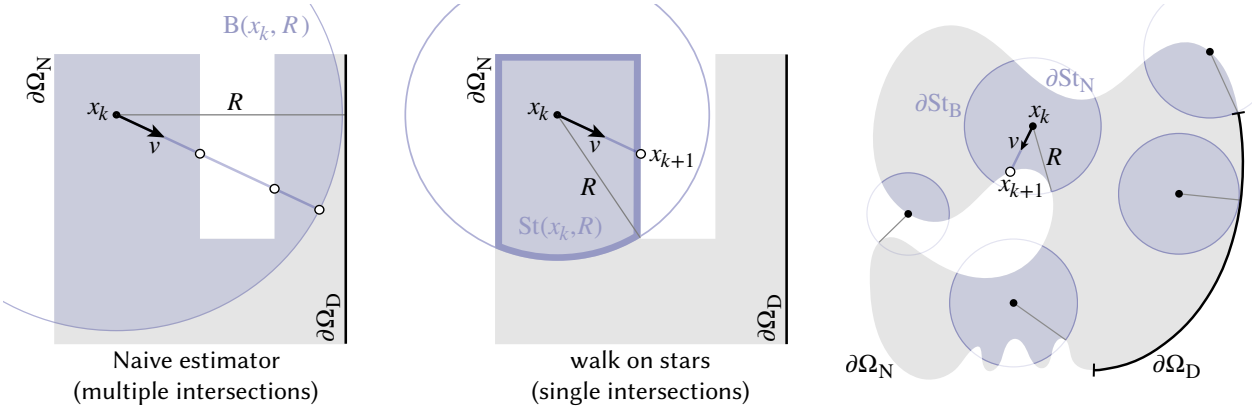


Figure 4.3: Left: For a ball $\mathrm{B}(x_k, R)$ *whose radius* R *is the distance to the Dirichlet boundary* $\partial\Omega_\mathrm{D}$*, the solution to a Poisson equation has to be estimated at all ray intersections, sampled proportional to signed solid angle. Center and right: WoSt instead restricts* $\mathrm{B} \cap \Omega$ *to be star-shaped relative to* x_k *to avoid more than one intersection, and estimates the PDE solution at the first intersection point* x_{k+1} *on either* $\partial\mathrm{B}$ *inside* Ω*, or the Neumann boundary* $\partial\Omega_\mathrm{N}$ *inside* B*. The radius* R *of a star-shaped region* $\mathrm{St}(x_k, R)$ *is the minimum of the distance to the closest silhouette point on the Neumann boundary* $\partial\Omega_\mathrm{N}$ *and the closest point on the Dirichlet boundary* $\partial\Omega_\mathrm{D}$*.*

4.2.2 Random Walk On Star-Shaped Region

The next walk location is importance sampled from the Poisson kernel of a ball centered at the current point x_k, *i.e.*, $x_{k+1} \sim p^{\partial\mathrm{St}} = P^\mathrm{B}(x_k, x_{k+1})$. For a Poisson equation in $\mathbb{R}^3$, this kernel equals

$$P^\mathrm{B}_{3\mathrm{D}}(x_k, x_{k+1}) = \frac{n_{x_{k+1}} \cdot (x_{k+1} - x_k)}{4\pi \left\| x_{k+1} - x_k \right\|^3}. \tag{4.3}$$

We use the same sampling density for a screened Poisson equation, as its corresponding kernel simply multiplies P^B by a constant in $[0, 1)$ determined by the absorption coefficient (Eq. A.15).

Eq. 4.3 coincides with the *signed solid angle* subtended by $\partial\mathrm{St}$ at x_{k+1} with respect to x_k [12, 114]. In rendering, this term appears in the *light transport equation (LTE)* [203, Eq. 14.15]. Unlike the BIE, the LTE multiplies P^B with a binary *visibility* $V(x, y)$ that equals 1 if x and y are mutually visible. Visibility ensures the product $V(x_k, y) P^{\mathrm{B}(x_k, R)}(x_k, y)$ is nonnegative: positive if y is visible from x_k, and zero otherwise. Through a change of variables, this product can then be importance sampled via *directional sampling*, *i.e.*, cast a ray from x_k in a direction v uniformly sampled from the unit sphere, and find its first intersection with $\partial\mathrm{St}$:

$$x_{k+1} \coloneqq x_k + t_\partial v, \qquad t_\partial \coloneqq \min\{t \in [0, +\infty) : x_k + tv \in \partial\mathrm{St}(x_k, r)\}. \tag{4.4}$$

We refer to Veach [251] for further details on the relationship between area and directional sampling. If x_k lies on the boundary, then the ray origin should be offset slightly along the inward boundary normal to avoid self-intersections, as described in Wächter and Binder [257].

Non-Visible Regions

Since Brownian motion can effectively "walk around corners" (Fig. 2.7), the BIE has no visibility term. Hence, the solution value at x_k can depend on non-visible points x_{k+1}, complicating use of directional sampling. In particular, if the region around x_k is nonconvex, then a naïve strategy is to estimate u at *all* intersections along a ray from x_k (Fig. 4.3, *left*), yielding a branching walk that increases in size exponentially. One could instead use just a single randomly selected intersection,

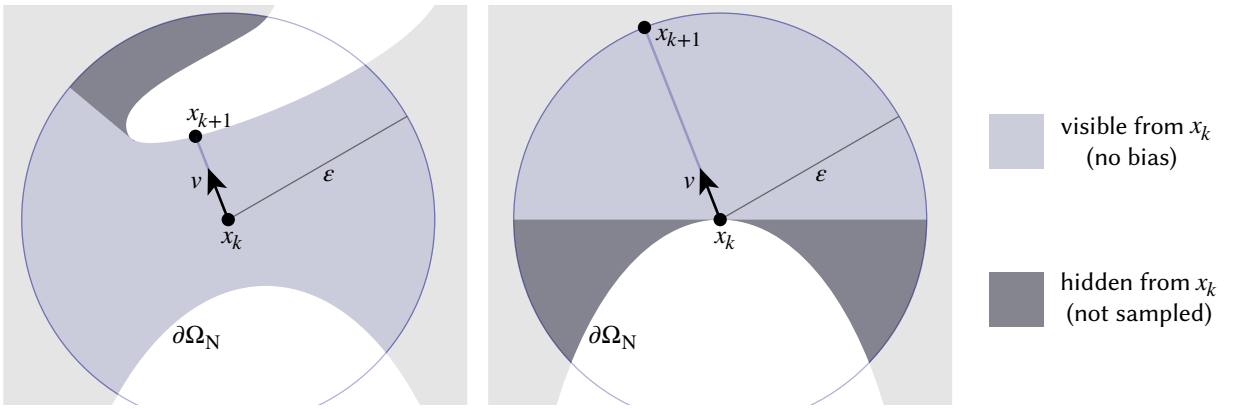


Figure 4.4: WoSt uses balls with radius no smaller than ε to prevent walks from stopping near concave Neumann boundaries. Left: We only sample parts of $\partial\Omega_N$ directly visible to x_k inside any ball $B(x_k, \varepsilon)$, implicitly assuming the function u is zero elsewhere. Right: Hemispherical boundary sampling ensures the next walk location x_{k+1} does not leave the domain, but also incurs a small bias in $\widehat{u}$ when x_k lies on a concave boundary.

but this approach yields extremely high variance (see Fig. 8.17 & 8.18), for two reasons. First, the recursive solution estimate must be multiplied by the number of intersections to account for the expected contribution from each intersection, causing a blowup in value as walk length increases. Second, the Poisson kernel alternates sign between consecutive intersection points along a ray, yielding unstable estimates due to cancellation [125, Ch. 4]. These issues are also the root cause of high variance in the *walk on boundary (WoB)* method [217, 242], which we compare against in Sec. 8.3.2. Moreover, using just the first intersection leads to a biased estimator, as the solution u is then effectively assumed to be zero on non-visible parts of the boundary.

Sampling Star-Shaped Regions

To avoid these issues, some past work assumes the *entire* Neumann boundary ∂St_N is convex [63, 235], yielding only one intersection for any region $B(x_k, R) \cap \Omega$ where $R := d_{\text{Dirichlet}}(x_k)$. This assumption of course limits the applicability of such estimators.

We instead let R be the minimum of the distance $d_{\text{Dirichlet}}(x_k)$ to the Dirichlet boundary, and the distance $d_{\text{silhouette}}(x_k)$ to the closest point on the visibility silhouette of $\partial\Omega_N$. The connected component of $B(x_k, R) \cap \Omega$ containing x_k then defines a star-shaped region $St(x_k, R)$. Fig. 4.3 (*center* & *right*) shows several examples. We can thus sample points on the region boundary ∂St by simply taking the first point along a ray from x_k that intersects either $\partial B(x_k, R)$ or $\partial\Omega_N$. Like the original WoS algorithm (and unlike the reflections in Fig. 3.5), WoSt can therefore take large steps when far from the Dirichlet boundary. Though other star-shaped sets could be also used, our approach is motivated by the fact that the closest silhouette point is easy to compute—as will be discussed in Sec. 6.2.1.

Epsilon Clamp For Star-Shaped Regions

Near concave parts of the Neumann boundary, the distance to the closest silhouette point on $\partial\Omega_N$ shrinks to zero (Fig. 4.5), stalling the progress of random walks. We hence limit the radius R used to define $St(x_k, R)$ to be greater than a user-defined ε, but still use only the first

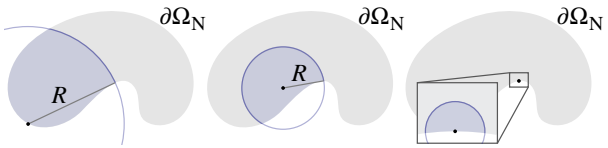


Figure 4.5: The distance to the visibility silhouette shrinks as a query point approaches a concave part on $\partial\Omega_N$.

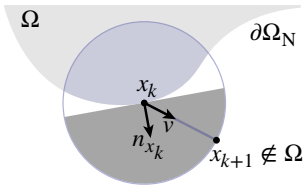
ALGORITHM 2: WalkOnStars(x, n_x, ε)
Note: Changes to WoS from Alg. 1 are annotated with comments in blue.

Input: Starting position $x \in \Omega$ of random walk, normal n_x at x (undefined if $x \notin \partial\Omega_N$), and ε-shell.
Output: Single-sample MC estimate $\widehat{u}(x)$ of Poisson equation with Dirichlet-Neumann conditions.

$d, \overline{x} \leftarrow$ ClosestPt$(\partial\Omega_D, x)$ ▷*Compute distance to absorbing boundary $\partial\Omega_D$ (Sec. 6.1)*
if $d < \varepsilon$ **then return** $g(\overline{x})$ ▷*Return boundary value g at closest pt $\overline{x}$ if $x \in \partial\Omega_D^\varepsilon$*
$R \leftarrow$ Max(StarRegionRadius$(\partial\Omega_N, x, d)$, ε) ▷*Compute star region St radius, $\varepsilon \leq R \leq d$ (Sec. 6.2.1)*
$v \leftarrow$ SampleUnitSphere() ▷*Sample direction v uniformly on unit sphere*
if $x \in \partial\Omega_N$ **and** $n_x \cdot v > 0$ **then** $v \leftarrow -v$ ▷*Ensure v is sampled on hemisphere with axis $-n_x$ if $x \in \partial\Omega_N$*
hit, $p, n_p \leftarrow$ Intersect$(\partial\Omega_N, x, v, R)$ ▷*Intersect $\partial\mathrm{St}_N$ with ray $x + Rv$, and get first hit*
if not hit **then** $p \leftarrow x + R\,v$ ▷*If there is no hit with $\partial\mathrm{St}_N$, set next walk position on $\partial\mathrm{St}_B$*
$\widehat{I}_h \leftarrow$ ReflectingBoundaryEstimate(x, R) ▷*Estimate contribution from boundary term h (Sec. 4.2.3)*
$\widehat{I}_f \leftarrow$ SourceEstimate(x, p, v, R) ▷*Estimate contribution from source term f (Sec. 4.2.4)*
return WalkOnStars$(p, n_p, \varepsilon) + \widehat{I}_h + \widehat{I}_f$ ▷*Repeat from next walk position p (n_p undefined if $x \notin \partial\Omega_N$)*

ray intersection to sample the next point x_{k+1}. This scheme incurs a small amount of bias when St is not star-shaped, since we assume the solution u is zero on any piece of ∂St not visible from x_k (Fig. 4.4, *left*). As with the epsilon parameter for the Dirichlet shell $\partial\Omega_D^\varepsilon$, a smaller ε value reduces bias near concave regions of $\partial\Omega_N$ at the expense of performance. We study this performance-bias tradeoff in Sec. 6.3. In practice our star-shaped regions tend to be much larger than ε, even slightly away from a concave boundary.

Hemispherical Sampling On The Neumann Boundary



When x_k lies on $\partial\Omega_N$, sampling v from the unit sphere can yield points x_{k+1} outside Ω (inset). Here we instead sample v from the hemisphere around the normal n_{x_k}. This scheme effectively invokes the *reflection principle* of Brownian motion [113], across the halfplane at the base of the hemisphere. A useful consequence is that the $\alpha(x_k) = 1/2$ in the denominator of the first term in Eq. 4.2 is canceled by the factor $1/2$ we get from sampling a hemisphere, rather than a sphere. This prevents our recursive estimator from picking up a multiplicative factor of two each time a walk reaches $\partial\Omega_N$. Note that if $\partial\Omega_N$ is concave at x_k, we again incur a small amount of bias (Fig. 4.4, *right*).

4.2.3 Sampling Neumann Boundary Conditions

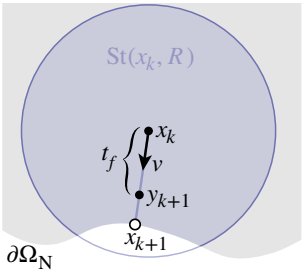
For problems with nonzero Neumann conditions, we must evaluate the second term in Eq. 4.2. We could in theory use x_{k+1} to estimate both boundary terms, however, this approach yields biased results, as direction sampling (Eq. 4.4) never samples a point x_{k+1} on a flat Neumann boundary when $x_k \in \partial\Omega_N$, even if h is non-zero there. This is because the Poisson kernel P^B in Eq. 4.3 is zero at points x_{k+1} where $n_{x_{k+1}} \perp (x_{k+1} - x_k)$. Moreover, even for non-flat boundaries, the ratio $G^B/p^{\partial\mathrm{St}_N} := G^B/P^B$ in the second term in Eq. 4.2 results in high-variance estimates, as the Poisson kernel can take on both very large and small values.

Instead, we sample a point z_{k+1} uniformly on the Neumann boundary $\partial\Omega_N$, and add a contribution $h(z_{k+1})$ only if z_{k+1} is also contained in $\partial\mathrm{St}_N$. This estimate remains unbiased, since we effectively integrate the same function (h restricted to $\partial\mathrm{St}_N$) over a larger domain. However, sampling the entire Neumann boundary can lead to high variance in the estimator, as most samples

will not lie on $\partial \mathrm{St}_{\mathrm{N}}$. Likewise, rejection sampling (Sec. 3.1.2) can be prohibitively expensive since $\partial \mathrm{St}_{\mathrm{N}}$ can be much smaller than $\partial\Omega_{\mathrm{N}}$. In Sec. 6.2.4, we hence describe a strategy for efficiently generating *visible* samples z_{k+1} close to x_k, which significantly reduces variance.

4.2.4 Sampling The Source Term

Finally, we sample a point $y_{k+1} \in \mathrm{B}(x_k, R)$ to evaluate the third term in Eq. 4.2. We reuse the ray direction v we sampled to generate x_{k+1} (Eq. 4.4), and set $y_{k+1} \coloneqq x_k + t_f v$, where we sample the distance t_f proportionally to G^{B} (App. A.1.2). If the sampled distance t_f is greater than $t_\partial \coloneqq \|x_{k+1} - x_k\|$, then the point y_{k+1} lies outside the star-shaped region $\mathrm{St}(x_k, R)$, and we reject it (inset). As in Sec. 4.2.2, (re)using a hemispherical direction cancels $\alpha(x_k) = 1/2$ when $x_k \in \partial\Omega_{\mathrm{N}}$.



4.2.5 Final Estimator

Our final WoSt estimator is defined recursively as

$$\widehat{u}(x_k) \coloneqq \begin{cases} g(\overline{x}_k), & \overline{x}_k \in \partial\Omega_{\mathrm{D}}^{\varepsilon}, \\ \widehat{u}(x_{k+1}) + \widehat{I}_h + \widehat{I}_f & \text{otherwise}, \end{cases} \tag{4.5}$$

where x_{k+1} is the next walk location in Ω or on $\partial\Omega_{\mathrm{N}}$, and $\widehat{I}_h$ and $\widehat{I}_f$ are non-recursive Neumann and source contributions, respectively. This estimator maintains the general structure of a WoS estimator, and thus introduces little implementation overhead. Alg. 2 provides pseudocode.

4.3 Walk On Stars With Robin Conditions

WoSt can take long walks when the boundary conditions are mostly Neumann: it must reflect off the Neumann boundary, and can terminate only on the Dirichlet boundary (Fig. 4.1, *center*). This is analogous to path tracing a scene where all materials have albedo one, such as a room of perfect mirrors. Here we extend WoSt to support Robin conditions, which provide greater physical realism in describing real-world thermal, electromagnetic, elastic and fluidic materials [86, 90, 94, 228] than purely absorbing or reflecting boundaries, corresponding to, *e.g.*, a blackbody or perfect insulator (respectively). Monte Carlo estimators for Robin problems are also, in general, more efficient than estimators for Neumann-dominated problems, as random walks simulating *partially reflecting Brownian motion* [86] can be absorbed on $\partial\Omega_{\mathrm{R}}$ (Fig. 2.7, *right*), whereas Neumann walks may require many steps to reach $\partial\Omega_{\mathrm{D}}$, resulting in high computation time.

To add support for Robin boundary to WoSt, our main modification is to change how we select the radius R of the balls $\mathrm{B}(x, R)$ used to form star-shaped regions, leaving the rest of the algorithm largely unchanged. As shown in Fig. 4.6, the Robin coefficient μ from Eq. 2.1 linearly interpolates between Neumann conditions ($\mu = 0$) and Dirichlet conditions ($\mu = \infty$), which means that as μ increases, a Robin boundary $\partial\Omega_{\mathrm{R}}$ becomes less reflecting and more absorbing. Intuitively, we expect μ to have a similar interpolatory impact on R: in Fig. 4.7, we show R achieves its maximal value R_0 on a boundary when $\mu = 0$ (*i.e.*, $\partial\Omega_{\mathrm{R}} \equiv \partial\Omega_{\mathrm{N}}$), with B expanding freely through $\partial\Omega_{\mathrm{N}}$ until it encounters a silhouette point. On the other hand, R equals its minimal value R_∞ when $\mu = \infty$ (*i.e.*, $\partial\Omega_{\mathrm{R}} \equiv \partial\Omega_{\mathrm{D}}$), as B cannot cross through $\partial\Omega_{\mathrm{D}}$. Otherwise, R transitions smoothly from R_0 to R_∞ as μ increases, with $\partial\Omega_{\mathrm{R}}$ becoming less "permeable" as B expands.

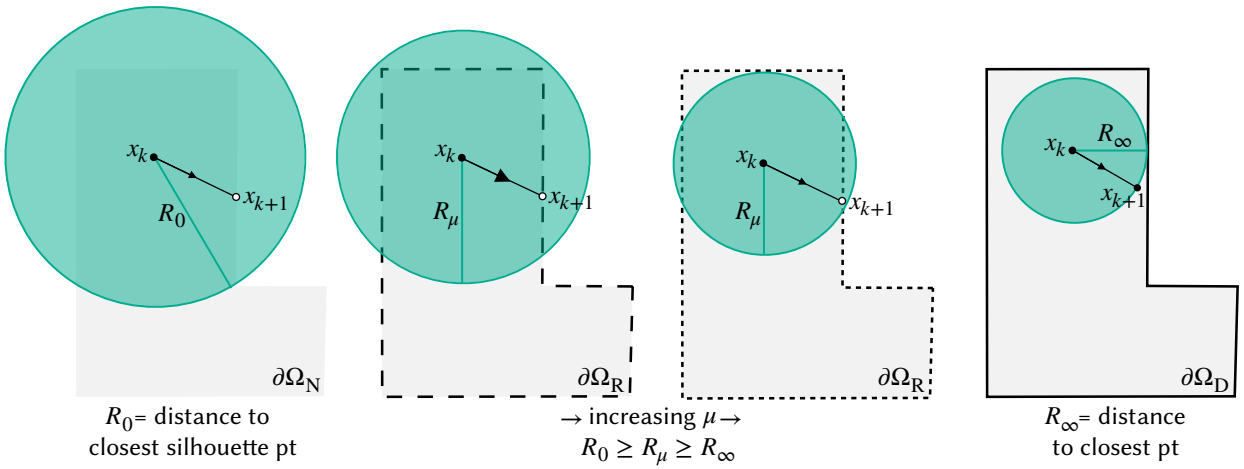


Figure 4.7: By increasing the coefficient μ in Eq. 2.1, the radius of a star-shaped region for a Robin boundary reduces continuously from the distance to the closest silhouette point on the Neumann boundary where $\mu = 0$, to the distance to the closest point on the Dirichlet boundary where $\mu = \infty$.

In the rest of this section, we formalize this intuition for the ball size in three steps: in Sec. 4.3.1 & 4.3.2, we introduce a *reflectance function* ρ_μ, which adds support for Robin conditions to the BIE we use to design our WoSt estimator. In Sec. 4.3.3, we explain why the introduction of ρ_μ necessitates selecting a radius R smaller than the one used with Neumann conditions, and how to use ρ_μ to facilitate this selection (Fig. 4.8). Finally in Sec. 4.3.4, we show how ρ_μ allows random walks to be terminated early through Russian roulette (Sec. 3.1.1) to improve efficiency (Fig. 4.6). Alg. 3 highlights these changes in green. Notably, this construction yields an estimator that demonstrates reliable Monte Carlo convergence with increasing number of walks for *any* combination of Dirichlet, Neumann and Robin conditions (Sec. 6.3). It also typically has orders of magnitude less error than other grid-free techniques for BVPs, like the walk on boundary method (Fig. 8.18).

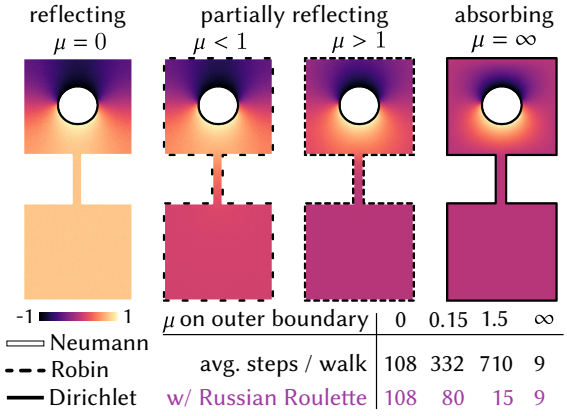


Figure 4.6: As μ increases and the boundary becomes less reflecting, Russian roulette helps make WoSt more efficient by probabilistically terminating walks on the Robin boundary, which reduces the average number of steps per walk.

4.3.1 Boundary Integral Formulation

To derive a boundary integral that accounts for Robin conditions, we follow largely the same derivation as that in Sec. 4.2. The main difference is that we use the *Brakhage-Werner trick* from potential theory [182] to substitute $\partial u/\partial n = \ell - \mu \cdot u$ (from the Robin condition in Eq. 2.1) on $\partial\mathrm{St}_R := \partial\Omega_R \cap \partial\mathrm{St}$. Rearranging terms then yields:

$$\alpha(x)\, u(x) = \int_{\partial\mathrm{St}(x,R)} \rho_\mu(x,z)\; P^B(x,z)\; u(z)\, \mathrm{d}z + \int_{\partial\mathrm{St}_R(x,R)} G^B(x,z)\; \ell(z)\, \mathrm{d}z \;+\; \int_{\mathrm{St}(x,R)} G^B(x,y)\; f(y)\, \mathrm{d}y, \tag{4.6}$$

where the spatially-varying function ρ_μ is defined as

$$\rho_\mu(x,z) \coloneqq \begin{cases} 1 - \mu(z)\frac{G^{\mathrm{B}}(x,z)}{P^{\mathrm{B}}(x,z)}, & \text{on } \partial\mathrm{St}_{\mathrm{R}}, \\ 1, & \text{on } \partial\mathrm{St}_{\mathrm{B}}. \end{cases} \tag{4.7}$$

Eq. 4.6 is nearly identical to Eq. 4.1 for Neumann problems, and even reduces to it when $\mu = 0$. We will treat Neumann conditions as special case of Robin moving forward. Importantly, u is the only (recursively-defined) unknown in this equation. Similar to Sec. 4.2.1, we can therefore use single-sample Monte Carlo to derive a recursive WoSt estimator with Robin conditions.

4.3.2 Monte Carlo Estimation

The Walk On Stars Estimator With Robin Conditions

A single-sample Monte Carlo estimator for Eq. 4.6 is given by

$$\begin{aligned} \widehat{u}(x_k) = &\frac{\rho_\mu(x_k, x_{k+1})\, P^{\mathrm{B}}(x_k, x_{k+1})\, \widehat{u}(x_{k+1})}{\alpha(x_k)\, p^{\partial\mathrm{St}(x_k,R)}(x_{k+1})} \\ &+ \frac{G^{\mathrm{B}}(x_k, z_{k+1})\, \ell(z_{k+1})}{\alpha(x_k)\, p^{\partial\mathrm{St}_{\mathrm{R}}(x_k,R)}(z_{k+1})} + \frac{G^{\mathrm{B}}(x_k, y_{k+1})\, f(y_{k+1})}{\alpha(x_k)\, p^{\mathrm{St}(x_k,R)}(y_{k+1})}. \end{aligned} \tag{4.8}$$

Given how similar the integrands are in Eq. 4.1 and 4.6, we can sample the points $x_{k+1} \in \partial\mathrm{St}$, $z_{k+1} \in \partial\mathrm{St}_{\mathrm{R}}$, $y_{k+1} \in \mathrm{St}$ using the same densities $p^{\partial\mathrm{St}}$, $p^{\partial\mathrm{St}_{\mathrm{R}}}$, p^{St} (respectively) as those in Sec. 4.2.1–4.2.4. Apart from the introduction of ρ_μ, the WoSt estimator with Robin conditions is unchanged from the estimator with Neumann conditions. Next, we show how to select a radius R for each star-shaped region $\mathrm{St}(x_k, R)$, and terminate random walks on $\partial\Omega_{\mathrm{R}}$—these are the only two places where estimation deviates from Sec. 4.2 (see Alg. 3). We treat Dirichlet conditions as separate from Robin for algorithmic convenience, even though they are a special case with $\mu = \infty$.

4.3.3 Using Reflectance To Select Ball Radius

As a first attempt, we could choose R to equal the distance $d_{\text{silhouette}}$ to the closest silhouette point on $\partial\Omega_{\mathrm{R}}$ from x, *i.e.*, the radius R_0 we use for Neumann problems (Fig. 4.8, *center*). To understand why this is a bad choice for R, we must consider the values $\rho_\mu(x,z)$ (Eq. 4.7) assumes on different parts of $\partial\mathrm{St}(x,R)$. In particular, irrespective of μ, $\rho_\mu(x,z) = 1$ at points $z \in \partial\mathrm{St}_{\mathrm{B}}$, since $G^{\mathrm{B}}(x,z) = 0$. When $\mu = 0$, $\rho_\mu(x,z)$ likewise simplifies to 1 at points $z \in \partial\mathrm{St}_{\mathrm{R}}$, recovering the original setup for Neumann problems. However, when $\mu > 0$, ρ_μ in general varies between $-\infty$ and 1 if we use $R = R_0 \equiv d_{\text{silhouette}}(x)$. This choice of radius then leads to extremely high variance in the *recursive* estimator in Eq. 4.8, for two reasons. First, the solution estimate $\widehat{u}$ accumulates a *throughput*[1] $\prod_k \rho_\mu(x_k, x_{k+1})$ that becomes unbounded in magnitude as walk length k increases. Second, the function ρ_μ, and thus throughput, can change sign along a walk, which results in unstable estimates due to cancellations [125, Ch. 4]. In App. C, we provide an operator-theoretic analysis of boundary integral equations to more formally explain the issues with naïve estimation of Eq. 4.6 and our solution to it, which we discuss below.

[1]Unlike the throughput of an estimator for a screened Poisson equation (Sec. 3.2.2), the throughput of a walk in this setting represents the probability with which Brownian motion is reflected off (and hence not absorbed on) $\partial\Omega_{\mathrm{R}}$.

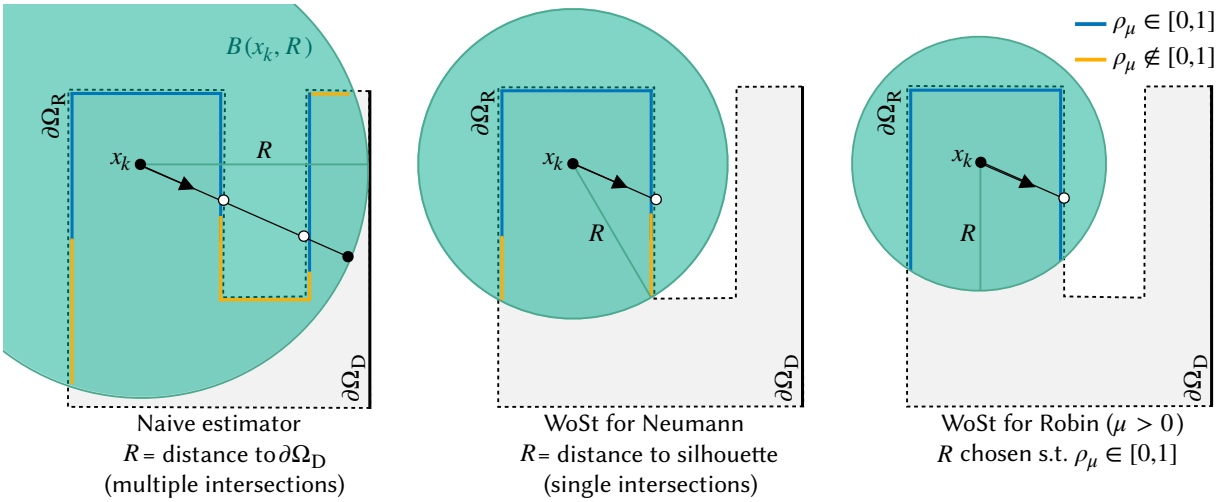


Figure 4.8: We use reflectance values ρ_μ on the Robin boundary $\partial\Omega_R$ to help determine the radius R of a star-shaped region $\mathrm{St}(x_k, R)$. Left: For a ball $\mathrm{B}(x_k, R)$ where R is the distance to the Dirichlet boundary $\partial\Omega_D$, the PDE solution must be estimated at all ray intersections sampled proportional to signed solid angle. Center: WoSt with Neumann conditions avoids multiple intersections by instead restricting $\mathrm{B} \cap \Omega$ to be star-shaped relative to x_k, and estimating the solution at a single intersection point $x_{k+1} \in \partial\mathrm{St}$. Right: For Robin conditions with $\mu > 0$, R is restricted further to ensure that ρ_μ has a value between 0 and 1.

Shrinking The Radius

To ensure throughput remains positive and bounded regardless of walk length, we choose a radius $R \leq R_0$ such that $\rho_\mu \in [0, 1]$. To achieve this, we substitute expressions for the 3D Green's function and Poisson kernel of a ball $\mathrm{B}(x, R)$ (App. A.1.2) into Eq. 4.7. For any point $z \in \partial\mathrm{St}_R$,

$$\rho_\mu(x, z) = 1 - \frac{\mu(z)\, r}{\cos\theta}\left(1 - \frac{r}{R}\right), \tag{4.9}$$

where $r = \|z - x\|$ and $\cos\theta = {}^{n_z \cdot (z-x)}/_{r}$. The terms $1 - {}^{r}/_{R}$ and $\cos\theta$ are both positive, since $r \leq R$ and $\partial\mathrm{St}_R$ is front-facing by construction (as St is star-shaped). To restrict $\rho_\mu \in [0, 1]$, we therefore require

$$\frac{\mu(z)\, r}{\cos\theta}\left(1 - \frac{r}{R}\right) \leq 1. \tag{4.10}$$

Rearranging terms then gives an upper bound on the radius R,

$$R \leq \frac{r}{1 - \frac{\cos\theta}{\mu(z)\, r}} \quad \text{when } r > \frac{\cos\theta}{\mu(z)}, \tag{4.11}$$

which must hold at *all* points $z \in \partial\mathrm{St}_R$ (Fig. 4.8, *right*). When $r \leq {}^{\cos\theta}/_{\mu(z)}$, $\rho_\mu \in [0, 1]$ for any $r < R$; in this case we set R equal to the distance $d_{\text{Dirichlet}}(x)$ to $\partial\Omega_D$, or ∞ when $\partial\Omega_D = \emptyset$.

As μ increases from 0 to ∞ on $\partial\Omega_R$, the bound in Eq. 4.11 reduces continuously from $R_0 \equiv d_{\text{silhouette}}(x)$ to $R_\infty \equiv d_{\text{Dirichlet}}(x)$. It asymptotically recovers the radii R_0 and R_∞ WoSt uses for pure Neumann and Dirichlet conditions as μ approaches 0 and ∞ (respectively). With this bound for the star-shaped region radius R, we call ρ_μ the *reflectance* for the Robin boundary inside $\mathrm{St}(x, R)$, as it encodes the probability with which a random walk is reflected off $\partial\mathrm{St}_R$. In Sec. 6.2.2, we describe how to compute R efficiently on triangle meshes, with only small modifications to how $d_{\text{silhouette}}$ is computed for $\partial\Omega_N$ (Sec. 6.2.1). App. D provides corresponding expressions for the reflectance and radius bound in 2D domains.

ALGORITHM 3: WalkOnStars(x, n_x, ε)
Note: Changes to WoSt from Alg. 2 are annotated with comments in green.

Input: Starting position $x \in \Omega$ of random walk, normal n_x at x (undefined if $x \notin \partial\Omega_R$), and ε-shell.
Output: Single-sample MC estimate $\widehat{u}(x)$ of Poisson equation with Dirichlet-Robin conditions.

1: $d, \overline{x} \leftarrow$ ClosestPt$(\partial\Omega_D, x)$ ▷*Compute distance to absorbing boundary $\partial\Omega_D$ (Sec. 6.1)*
2: **if** $d < \varepsilon$ **then return** $g(\overline{x})$ ▷*Return boundary value g at closest pt $\overline{x}$ if $x \in \partial\Omega_D^\varepsilon$*
3: $R \leftarrow$ Max(StarRegionRadius$(\partial\Omega_R, x, d)$, ε) ▷*Compute star region St radius, $\varepsilon \leq R \leq d$ (Sec. 6.2.2)*
4: $v \leftarrow$ SampleUnitSphere() ▷*Sample direction v uniformly on unit sphere*
5: **if** $x \in \partial\Omega_R$ **and** $n_x \cdot v > 0$ **then** $v \leftarrow -v$ ▷*Ensure v is sampled on hemisphere with axis $-n_x$ if $x \in \partial\Omega_R$*
6: hit, $p, n_p \leftarrow$ Intersect$(\partial\Omega_R, x, v, R)$ ▷*Intersect $\partial\mathrm{St}_R$ with ray $x + Rv$, and get first hit*
7: **if not** hit **then** $p \leftarrow x + R\,v$ ▷*If there is no hit with $\partial\mathrm{St}_R$, set next walk position on $\partial\mathrm{St}_B$*
8: $\widehat{I}_\ell \leftarrow$ ReflectingBoundaryEstimate(x, R) ▷*Estimate contribution from boundary term ℓ (Sec. 4.2.3)*
9: $\widehat{I}_f \leftarrow$ SourceEstimate(x, p, v, R) ▷*Estimate contribution from source term f (Sec. 4.2.4)*
10: $\rho_\mu \leftarrow$ Clamp$(1 - \mu(p)G^{B(x,R)}(x,p)/P^{B(x,R)}(x,p),\ 0,\ 1)$ ▷*Compute and clamp reflectance to $[0,1]$*
11: **if** $\rho_\mu <$ SampleUniform(0, 1) **then return** $\widehat{I}_\ell + \widehat{I}_f$ ▷*Attempt to terminate walk using Russian roulette*
12: **return** WalkOnStars$(p, n_p, \varepsilon) + \widehat{I}_\ell + \widehat{I}_f$ ▷*Repeat from next walk position p (n_p undefined if $x \notin \partial\Omega_R$)*

Epsilon Clamp

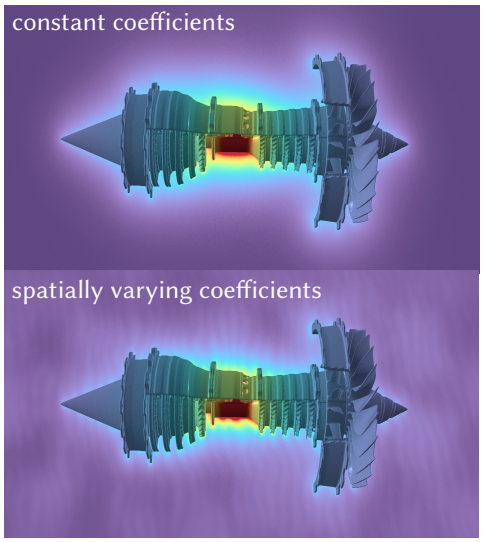
Irrespective of the value of μ, the radius R of a star-shaped region $\mathrm{St}(x_k, R)$ shrinks to zero as x_k approaches concave parts of $\partial\Omega_R$ (Fig. 4.5). As in Sec. 4.2.2, we clamp $R \geq \varepsilon$ to prevent walks from stalling near $\partial\Omega_R$. As $\mathrm{St}(x_k, \varepsilon)$ may no longer be star-shaped (Fig. 4.4), we additionally clamp ρ_μ to $[0, 1]$ to ensure throughput remains bounded (Alg. 3, *line* 10). This clamping is justified by the fact that as we make ε (and thus St) smaller, the value of ρ_μ automatically approaches 1. In Fig. 6.6, we use different ε values to empirically study the impact clamping has on bias and performance.

4.3.4 Using Russian Roulette To Terminate Walks

Using direction sampling (Eq. 4.4) to select the next walk location x_{k+1} perfectly importance samples (and hence cancels out) the Poisson kernel $P^B(x_k, x_{k+1})$ in Eq. 4.8, but not the reflectance $\rho_\mu(x_k, x_{k+1})$. As our choice of R guarantees $\rho_\mu(x_k, x_{k+1}) \in [0, 1]$, we can also cancel out this term using Russian roulette: we terminate walks at step k with probability $1 - \rho_\mu(x_k, x_{k+1})$, and cancel out the contribution $\rho_\mu(x_k, x_{k+1})$ in the walks that survive (Alg. 3, *line* 11). Using Russian roulette allows us to maintain a constant throughput of 1 in our walks and terminate them early, instead of waiting for walks to reach $\partial\Omega_D^\varepsilon$ while their throughput continues to shrink. This often leads to large efficiency gains, as we show in Fig. 4.6. We note that Russian roulette is not possible with Neumann conditions, where reflectance always equals 1. In this case, a walk must continue until it reaches $\partial\Omega_D^\varepsilon$. Otherwise it never terminates when $\partial\Omega_D = \emptyset$, which necessitates using Tikhonov regularization (Sec. 3.2.6).

Chapter 5

Estimators For PDEs With Spatially Varying Coefficients



In this chapter, we directly resolve detailed effects of spatially varying material densities (inset)—without resorting to homogenization of PDE coefficients (Fig. 5.1). WoS has previously been applied to a limited set of problems with piecewise constant coefficients [148, 155]; ours is the first grid-free method [223] for fairly general continuously-varying coefficients. Though the Monte Carlo estimators we have developed till now cannot handle the variable coefficients $\kappa(x)$, $\vec{\omega}(x)$ and $\sigma(x)$ from Sec. 2.1.1, they can solve PDEs with a variable source term $f(x)$. We hence apply a series of transformations (Fig. 5.2) that convert a variable-coefficient PDE such as Eq. 2.4 into an equivalent constant-coefficient screened Poisson equation (Eq. 2.3); we then use the integral version of screened Poisson (Eq. 2.15) to design modified walk on spheres algorithms. From a stochastic perspective, these transformations are equivalent to writing the Feynman–Kac formula purely in terms of Brownian motion, rather than a generic diffusion process (Sec. 2.3).

Our method applies whenever the PDE

$$\begin{aligned} \nabla \cdot (\kappa(x) \nabla u(x)) - \sigma(x) u(x) &= -f(x) \quad \text{on } \Omega, \\ u(x) &= g(x) \quad \text{on } \partial\Omega_{\mathrm{D}} \end{aligned} \tag{5.1}$$

is elliptic—which holds if the diffusion coefficient $\kappa(x)$ is strictly positive and the absorption coefficient $\sigma(x)$ is nonnegative [64, 72]. For brevity, we omit the drift term $\vec{\omega}(x) \cdot \nabla u(x)$ in Eq. 2.4 here, though the approach remains unchanged for equations with drift (see App. E). We focus on Dirichlet boundary conditions, and discuss extensions to reflecting conditions in Sec. 5.3.2. For readers not interested in the derivation, Eq. 5.9 gives the final integral formulation of Eq. 5.1 and Sec. 5.2 discusses algorithms, which we will evaluate in Ch. 8.

5.1 Transformations

Second order. We first expand the 2nd order term $\nabla \cdot (\kappa(x) \nabla u)$ in Eq. 5.1 via the product rule. We then divide the resulting equation by $\kappa(x)$, and apply the identity $\nabla \ln(\kappa(x)) = \nabla \kappa(x) / \kappa(x)$ to get

$$\Delta u(x) + \nabla \ln(\kappa(x)) \cdot \nabla u(x) - \frac{\sigma(x)}{\kappa(x)} u(x) = -\frac{f(x)}{\kappa(x)}. \tag{5.2}$$

At this point the 2nd order term Δu no longer has variable coefficients, but spatial variation in the lower order terms remains.

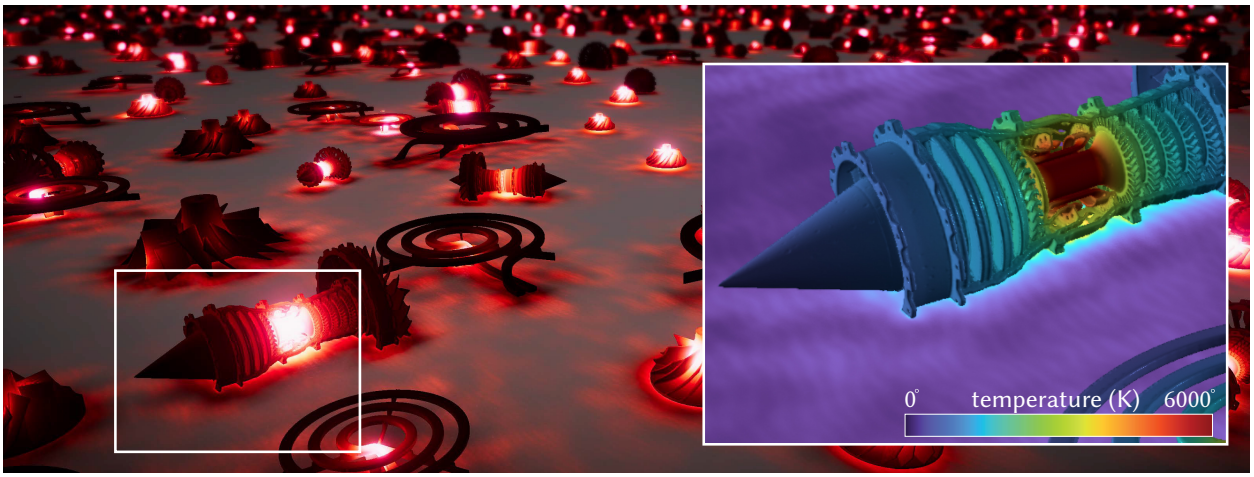


Figure 5.1: Distribution of heat (inset) radiating from infinitely many blackbodies—about 600M effective boundary vertices are visible from this viewpoint alone (we visualize a 2D slice of the full 3D solution). Our Monte Carlo PDE solver (Sec. 5.2.1) directly captures fine geometric detail and intricate spatially varying coefficients without volumetric meshing, sampling, or homogenizing the 3D domain, by building on techniques from volume rendering.

First order. A *Girsanov transformation* re-expresses a random process under a change of probability measure, *e.g.*, from a generic diffusion process X_t to an ordinary Brownian motion W_t [191, Ch. 8]. As shown in App. E, applying this transformation to Eq. 5.2 eliminates the 1st order operator from Eq. 5.2, shifting all spatial variation into the 0th order term:

$$\begin{aligned} \Delta U(x) - \sigma'(x) U(x) &= -f'(x) \quad \text{on } \Omega, \\ U(x) &= g'(x) \quad \text{on } \partial\Omega_{\mathrm{D}}. \end{aligned} \tag{5.3}$$

Here,

$$U(x) \coloneqq \sqrt{\kappa(x)}\, u(x), \quad g'(x) \coloneqq \sqrt{\kappa(x)}\, g(x), \quad f'(x) \coloneqq \frac{\sqrt{\kappa(x)}}{\kappa(x)}\, f(x),$$
$$\text{and} \quad \sigma'(x) \coloneqq \frac{\sigma(x)}{\kappa(x)} + \frac{1}{2}\left(\frac{\Delta\kappa(x)}{\kappa(x)} - \frac{|\nabla \ln(\kappa(x))|^2}{2}\right).$$

Equation 5.3 is equivalent to our original PDE with variable coefficients in Eq. 5.1, which can be verified by substituting the expressions for U, g', f' and σ' back into this equation.

Unlike the Feynman–Kac formula in Eq. 2.31 which involves a diffusion process X_t, the stochastic formula for Eq. 5.3 uses only simple Brownian motion W_t:

$$U(x) = \mathbb{E}\left[\mathrm{e}^{-\int_0^\tau \sigma'(W_t)\,\mathrm{d}t} g'(W_\tau) \;+\; \int_0^\tau \mathrm{e}^{-\int_0^t \sigma'(W_s)\,\mathrm{d}s} f'(W_t)\,\mathrm{d}t\right]. \tag{5.4}$$

Zeroth Order. The only remaining term on the left-hand side of Eq. 5.3 with a spatially varying coefficient is the 0th order screening term $\sigma'(x)U$. We hence apply a transformation inspired by delta tracking (Eq. 2.35) to shift this heterogeneity to a source term on the right-hand side. In doing so, we introduce a coefficient $\overline{\sigma} > 0$ by subtracting $\overline{\sigma}U$ from both sides of Eq. 5.3. The result is a PDE with the same basic form as a screened Poisson equation:

$$\begin{aligned} \Delta U(x) - \overline{\sigma} U(x) &= -f'(x,\, U) \quad \text{on } \Omega, \\ U(x) &= g'(x) \qquad \text{on } \partial\Omega_{\mathrm{D}}, \end{aligned} \tag{5.5}$$

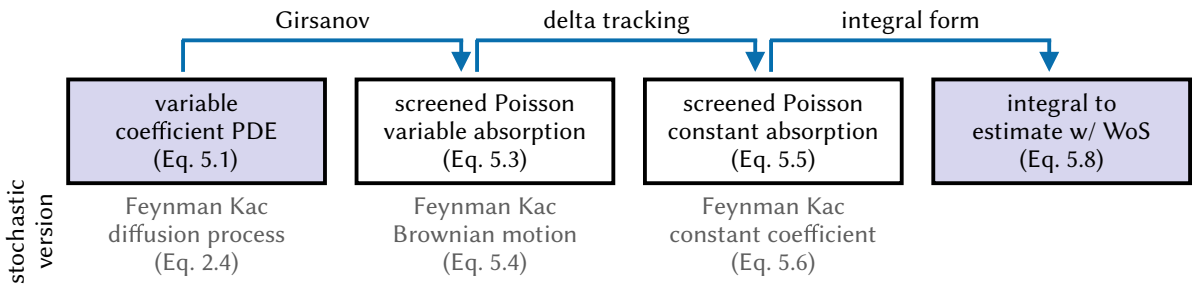


Figure 5.2: An overview of the transformations we apply to the variable-coefficient PDE in Eq. 5.1 to derive an integral formulation amenable to Monte Carlo estimation with WoS.

where

$$f'(x,\ U) := f(x) + (\overline{\sigma} - \sigma'(x))U(x). \tag{5.6}$$

Though only constant coefficients now appear on the left-hand side, *no approximation of any kind has been introduced*. Unlike a typical linear PDE however, the solution U appears on the right-hand side. As in volume rendering, we will account for this recursive dependence by applying recursive Monte Carlo estimation (Sec. 5.1.2)—a strategy not available in the traditional setting of, *e.g.*, finite element methods.

Like the transformed VRE in Eq. 2.36 and the stochastic formulas in Eq. 2.30, the stochastic expression for Eq. 5.5 also has a transmittance function $\mathrm{e}^{-\overline{\sigma}t}$ that no longer varies spatially:

$$U(x) = \mathbb{E}\left[\mathrm{e}^{-\overline{\sigma}\tau} g'(W_\tau) \ + \ \int_0^\tau \mathrm{e}^{-\overline{\sigma}t} f'(W_t, U)\,\mathrm{d}t\right]. \tag{5.7}$$

5.1.1 Integral Representation

We can now express the solution to Eq. 5.5 using the integral form of the constant coefficient screened Poisson equation (Eq. 2.15):

$$U(x) = \int_{\partial \mathrm{B}(c,R)} P^{\overline{\sigma},\mathrm{B}}(x,z)\ U(z)\,\mathrm{d}z \ + \ \int_{\mathrm{B}(c,R)} G^{\overline{\sigma},\mathrm{B}}(x,y)\ f'(y,U)\,\mathrm{d}y. \tag{5.8}$$

Recall that x can be any point inside the ball $\mathrm{B}(c,R)$—not just its center. Finally, we make the substitution $U(x) = \sqrt{\kappa(x)}\ u(x)$ from Eq. 5.3 to write this integral in terms of the original function u:

$$u(x) = \frac{1}{\sqrt{\kappa(x)}}\left(\int_{\partial \mathrm{B}(c,R)} P^{\overline{\sigma},\mathrm{B}}(x,z)\ \sqrt{\kappa(z)}\ u(z)\,\mathrm{d}z \ + \ \int_{\mathrm{B}(c,R)} G^{\overline{\sigma},\mathrm{B}}(x,y)\ f'(y,\sqrt{\kappa}\ u)\,\mathrm{d}y\right). \tag{5.9}$$

Unlike Eq. 2.15, we now have an integral equation that has unknown solution values u both in its boundary and volume terms. We will evaluate this integral via recursive application of Monte Carlo integration.

5.1.2 Monte Carlo Estimator

A single-sample estimator for Eq. 5.9 at a point $x_k \in \mathrm{B}(c,R)$ is given by

$$\widehat{u}(x_k) := \frac{1}{\sqrt{\kappa(x_k)}}\left(\overbrace{\frac{P^{\overline{\sigma},\mathrm{B}}(x_k,x_{k+1})\ \sqrt{\kappa(x_{k+1})}\ \widehat{u}(x_{k+1})}{p^{\partial\mathrm{B}(c,R)}(x_{k+1})\ \mathbb{P}^{\partial\mathrm{B}(c,R)}}}^{\text{evaluate boundary term with probability } \mathbb{P}^{\partial\mathrm{B}}} + \overbrace{\frac{G^{\overline{\sigma},\mathrm{B}}(x_k,y_{k+1})\ f'(y_{k+1},\sqrt{\kappa}\ \widehat{u})}{p^{\mathrm{B}(c,R)}(y_{k+1})\ \mathbb{P}^{\mathrm{B}(c,R)}}}^{\text{evaluate volume term with probability } \mathbb{P}^{\mathrm{B}}}\right), \tag{5.10}$$

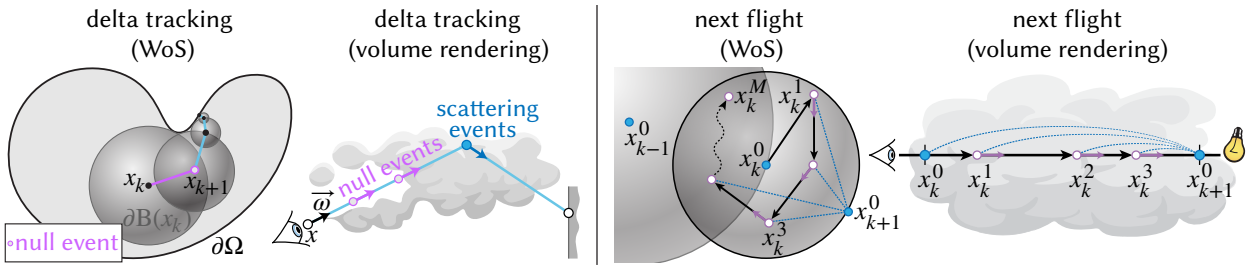
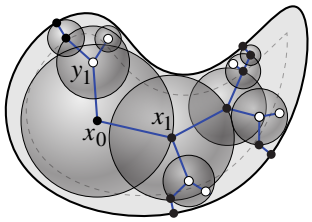


Figure 5.3: Left: Unlike WoS for constant coefficient problems, the delta tracking variant jumps to a random point either inside or on the boundary of the largest ball $B(x_k)$ centered at x_k. As in volume rendering (center left), null-events sampled inside B re-weight the solution estimate to account for spatial variations in the PDE coefficients. Center right: The next flight variant of WoS resolves variability in PDE coefficients by evaluating off-centered versions of the Green's function and Poisson kernel of a ball $B(x_k^l)$ in a random walk. Any walk inside B terminates at a point $x_{k+1}^0 \in \partial B(x_k^0)$ to avoid branching, similar to the next flight method for volume rendering which estimates transmittance along a ray with a predetermined endpoint (right).

where x_{k+1} and y_{k+1} are points sampled on the surface of and inside $B(c, R)$ according to probability densities $p^{\partial B}$ and p^B (respectively). Values $\mathbb{P}^{\partial B}, \mathbb{P}^B \in (0, 1]$ control the probability of sampling the boundary and volume terms. Letting $\mathbb{P}^{\partial B} = \mathbb{P}^B = 1$ yields exponential growth in the number of steps, since each walk branches into two (see inset), and walks do not terminate until both x_{k+1} and y_{k+1} are contained in $\partial\Omega_D^\varepsilon$. In the next section, we develop two WoS algorithms that avoid branching via a careful choice of $\mathbb{P}^{\partial B}$ and $\mathbb{P}^B$.



5.2 Algorithms

Due to the diversity of heterogeneous phenomena in nature, different algorithms for solving the volume rendering equation (Eq. 2.34) adopt different strategies to trade off between variance, bias, and computational cost [188]. Likewise, an algorithm for solving elliptic PDEs will be more effective when it is well-matched to the way coefficients are distributed in space. We provide a unified framework, based on Eq. 5.10, which enables us to explore variants of WoS appropriate for different problems—akin to the unidirectional estimator in Georgiev et al. [77, Eq. 14]. In particular, we devise two estimators inspired by the *delta tracking* [207, 270] and *next flight* [44] methods from volume rendering (Fig. 5.3). We also describe how to estimate the spatial gradient of the solution to Eq. 5.1 in Sec. 5.3.1.

5.2.1 Delta Tracking Variant Of Walk On Spheres

To avoid branching, our delta tracking variant of WoS uses a special property of the Poisson kernel $P^{\overline{\sigma},B}$ of a screened Poisson equation when x_k is at the ball center. Assuming $\overline{\sigma} > 0$, and letting $|G^{\overline{\sigma},B}(x)|$ be the integral of $G^{\overline{\sigma},B}$ over $B(x, R)$ (Eq. A.14), we have

$$P^{\overline{\sigma},B}(x_k, x_{k+1}) = \frac{1 - \overline{\sigma}|G^{\overline{\sigma},B}(x_k)|}{|\partial B(x_k, R)|}. \tag{5.11}$$

Since $\overline{\sigma}|G^{\overline{\sigma}}(x_k)| \in (0, 1)$ (see Eq. A.18), we can sample the boundary and volume terms with probabilities $\mathbb{P}^{\partial B} := 1 - \overline{\sigma}|G^{\overline{\sigma},B}(x_k)|$ and $\mathbb{P}^B := 1 - \mathbb{P}^{\partial B}$, yielding a non-branching estimator:

ALGORITHM 4: DeltaTrackingWoS(x, ε)

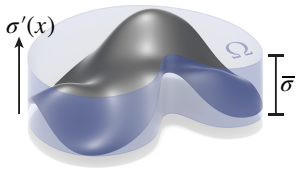
Input: Starting position $x \in \Omega$ of random walk, and ε-shell.
Output: Single-sample MC estimate $\widehat{u}(x)$ of Eq. 5.1 with Dirichlet conditions.

$d, \overline{x} \leftarrow$ ClosestPt$(\partial\Omega_{\mathrm{D}}, x)$ ▷*Compute distance to absorbing boundary $\partial\Omega_{\mathrm{D}}$ (Sec. 6.1)*
if $d < \varepsilon$ **then return** $g(\overline{x})$ ▷*Return boundary value g at closest pt $\overline{x}$ if $x \in \partial\Omega_{\mathrm{D}}^{\varepsilon}$*
$y \sim G^{\overline{\sigma},\mathrm{B}}(x, y)/|G^{\overline{\sigma},\mathrm{B}}(x)|$ ▷*Sample point y from $G^{\sigma,\mathrm{B}}$ inside ball $\mathrm{B}(x, d)$ (Sec. A.2.2)*
$\widehat{I}_f \leftarrow |G^{\overline{\sigma},\mathrm{B}}(x)| \cdot f(y)/\sqrt{\kappa(x)\kappa(y)}$ ▷*Estimate contribution from source term f (Eq. 5.3)*
if SampleUniform(0, 1) $\leq \overline{\sigma}|G^{\overline{\sigma},\mathrm{B}}(x)|$ **then** ▷*Sample volume term*
    return $\sqrt{\frac{\kappa(y)}{\kappa(x)}} \cdot \left(1 - \frac{\sigma'(y)}{\overline{\sigma}}\right) \cdot$ DeltaTrackingWoS$(y, \varepsilon) + \widehat{I}_f$ ▷*Repeat from next walk position y*
else ▷*Sample boundary term*
    $z \sim 1/|\partial\mathrm{B}(x, d)|$ ▷*Sample point z uniformly on $\partial\mathrm{B}(x, d)$*
    return $\sqrt{\frac{\kappa(z)}{\kappa(x)}} \cdot$ DeltaTrackingWoS$(z, \varepsilon) + \widehat{I}_f$ ▷*Repeat from next walk position z*

The Delta Tracking Estimator

$$\widehat{u}(x_k) := \begin{cases} g(\overline{x}_k), & \overline{x}_k \in \partial\Omega_{\mathrm{D}}^{\varepsilon}, \\ \frac{1}{\overline{\sigma}\sqrt{\kappa(x_k)}}\, f'(y_{k+1}, \sqrt{\kappa}\,\widehat{u}), & \mathrm{u} \sim \mathscr{U}(0,1) \leq \overline{\sigma}|G^{\overline{\sigma},\mathrm{B}}(x_k)|, \\ \sqrt{\kappa(x_{k+1})/\kappa(x_k)}\,\widehat{u}(x_{k+1}) & \text{otherwise.} \end{cases} \tag{5.12}$$

This estimator importance samples x_{k+1} and y_{k+1} via the densities $p^{\partial\mathrm{B}} := 1/|\partial\mathrm{B}(x_k, R)|$ and $p^{\mathrm{B}} := G^{\overline{\sigma},\mathrm{B}}(x_k, y_{k+1})/|G^{\overline{\sigma},\mathrm{B}}(x_k)|$, respectively. Use of constant-coefficient kernels $P^{\overline{\sigma},\mathrm{B}}$ and $G^{\overline{\sigma},\mathrm{B}}$ is critical, as kernels for the varying coefficient $\sigma'(x)$ from Eq. 5.3 are not known in closed form. However, spatial variation in $\sigma'(x)$ is still accounted for by the recursive source term f' (Eq. 5.6), which corresponds to probabilistically sampling null-events (Fig. 5.3, *left*). Alg. 4 provides pseudocode.



The coefficient $\overline{\sigma}$ is the only free parameter in this algorithm, and must be strictly positive to ensure that $\mathbb{P}^{\mathrm{B}} > 0$. In volume rendering one typically lets $\overline{\sigma} = \max(\sigma(x))$, which enables closed-form sampling of volumetric events (absorption, scattering, or null scattering) and boundary reflections. We instead let $\overline{\sigma} = \max(\sigma'(x)) - \min(\sigma'(x))$, since in general $\sigma'(x)$ can have both positive and negative values at different points $x \in \Omega$. More recent volume rendering research treats $\overline{\sigma}$ as a control variate rather than a bound [77, 188], to reduce variance based on the profile of the coefficients. In conjunction with clever choices for $p^{\partial\mathrm{B}}$ and p^{B}, these strategies can be more efficient than delta tracking; we leave such extensions to future work.

5.2.2 Next Flight Variant Of Walk On Spheres

The delta tracking variant of WoS takes more steps as $\overline{\sigma}$ increases (Fig. 5.4, *left*), since the Green's function becomes more localized (Fig. 2.4) and it becomes more likely that we sample a point in the volume than on the boundary ($\mathbb{P}^{\mathrm{B}} > \mathbb{P}^{\partial\mathrm{B}}$). Longer walks are ultimately more expensive, as distance queries are usually the bottleneck for WoS (much like ray intersections in path tracing).

We hence propose a variant based on the *next flight* scheme of Cramer [44], which takes big steps even when $\overline{\sigma}$ is large (Fig. 5.3, *right*). As usual we walk along points $x_0^0, x_1^0, \ldots$ sampled from successive spheres $\partial B(x_{k-1}, R)$, always estimating both boundary and volume terms ($\mathbb{P}^{\partial\mathrm{B}} = \mathbb{P}^{\mathrm{B}} = 1$). But rather than start a new walk to the boundary for the volume term, we take a "short

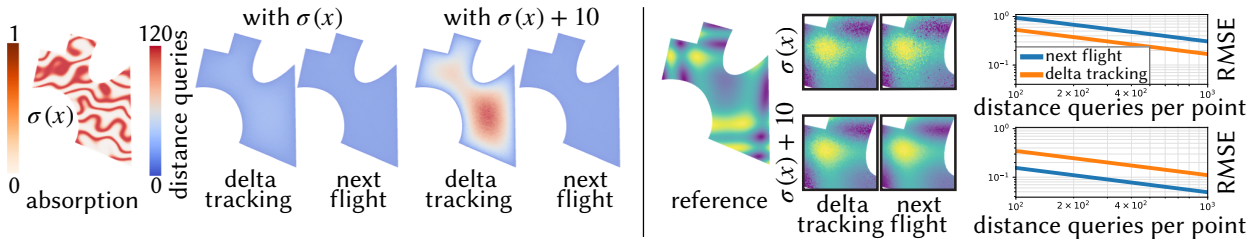


Figure 5.4: Left: As $\overline{\sigma}$ increases, the delta tracking variant of WoS requires more distance queries which reduces run-time performance, while the number of queries for the next flight strategy remain unchanged. Right: For smaller coefficients $\sigma(x)$, next flight exhibits higher variance due to greater correlation among samples. For larger coefficients $\sigma(x) + 10$, delta tracking exhibits more variance at equal compute time due to more distance queries.

off-center walk" $x_k^1, x_k^2, \ldots, x_k^M$ within each ball $\mathrm{B}(x_k^0, R)$, and re-use the estimate of the boundary contribution at x_{k+1}^0 for all steps in this short walk. An expression for this estimator is obtained by recursively expanding the definition of $\widehat{u}$ in the volume term of Eq. 5.10:

The Next Flight Estimator

$$\widehat{u}(x_k^0) := \frac{1}{\sqrt{\kappa(x_k^0)}} \left(\sqrt{\kappa(x_{k+1}^0)}\, \widehat{u}(x_{k+1}^0)\, \widehat{T}(x_k^0, x_{k+1}^0) \;+\; \widehat{I}_f(x_k^0) \right),$$

$$\text{where} \quad \widehat{T}(x_k^0, z) := \sum_{j=0}^{M} \frac{P^{\overline{\sigma},\mathrm{B}}(x_k^j, z)}{p^{\partial\mathrm{B}}(z)} \prod_{l=0}^{j-1} W(l),$$

$$\widehat{I}_f(x_k^0) := \sum_{j=1}^{M} \frac{f(x_k^j)}{\sqrt{\kappa(x_k^j)}\,(\overline{\sigma} - \sigma'(x_k^j))} \prod_{l=0}^{j-1} W(l),$$

$$W(l) := \frac{G^{\overline{\sigma},\mathrm{B}}(x_k^l, x_k^{l+1})\,(\overline{\sigma} - \sigma'(x_k^{l+1}))}{p^{\mathrm{B}}(x_k^{l+1})}. \tag{5.13}$$

For each point x_k^l, the subscript k indexes steps in the walk; the superscript l indexes points along the short walk in $\mathrm{B}(x_k^0, R)$. The number of terms M is determined by using the throughput $\prod_{l=0}^{j-1} W(l)$ as a probability for Russian roulette. Parameters $\overline{\sigma}$, p^{B} and $p^{\partial\mathrm{B}}$ are the same as in Sec. 5.2.1, but the Green's function $G^{\overline{\sigma},\mathrm{B}}$ and Poisson kernel $P^{\overline{\sigma},\mathrm{B}}$ must now be either evaluated or sampled using general off-centered formulas (App. A.2.2). Alg. 5 provides pseudocode.

A key benefit of next flight is that additional distance queries are not needed to evaluate $\widehat{T}$ and $\widehat{I}_f$ within $\mathrm{B}(x_k, R)$. However, decreased computation comes at the cost of increased correlation and variance in $\widehat{T}$ from reuse of $\widehat{u}(x_{k+1})$ and evaluation of off-centered kernels; see Fig. 5.4 (*right*).

5.3 Extensions

5.3.1 Spatial Gradient

Applications often require computing not just the solution to a PDE, but also the spatial gradient of the solution. Fortunately, estimating the gradient of Eq. 5.9 adds virtually no cost on top of estimating the solution itself. In particular, either of our variable-coefficient WoS algorithms can

ALGORITHM 5: NextFlightWoS(x, ε)

Input: Starting position $x \in \Omega$ of random walk, and ε-shell.
Output: Single-sample MC estimate $\widehat{u}(x)$ of Eq. 5.1 with Dirichlet conditions.

$d, \overline{x} \leftarrow$ ClosestPt$(\partial\Omega_{\mathrm{D}}, x)$ ▷*Compute distance to absorbing boundary $\partial\Omega_{\mathrm{D}}$ (Sec. 6.1)*
if $d < \varepsilon$ **then return** $g(\overline{x})$ ▷*Return boundary value g at closest pt $\overline{x}$ if $x \in \partial\Omega_{\mathrm{D}}^{\varepsilon}$*
$\widehat{T} \leftarrow 0,\ \widehat{I}_f \leftarrow 0,\ W \leftarrow 1$ ▷*Initialize series from Eq. 5.13*
$z \sim 1/|\partial \mathrm{B}(x, d)|$ ▷*Sample point z uniformly on ball boundary $\partial \mathrm{B}(x, d)$*
$x^c \leftarrow x$ ▷*Initialize temporary variable x^c to track walk position*
while True **do** ▷*Perform "off-center walk" inside* B
  $\widehat{T} \leftarrow \widehat{T} + |\partial \mathrm{B}(x, d)| \cdot P^{\overline{\sigma}, \mathrm{B}}(x^c, z) \cdot W$ ▷*Accumulate boundary contribution*
  $\mathbb{P}^{\mathrm{RR}} \leftarrow \min(1, W)$ ▷*Compute survival probability (Sec. 3.1.1)*
  if $\mathbb{P}^{\mathrm{RR}} <$ SampleUniform(0, 1) **then break** ▷*Attempt to terminate walk using Russian roulette*
  $W \leftarrow W / \mathbb{P}^{\mathrm{RR}}$ ▷*Update path throughput with survival probability*
  $x^l \sim 1/|\mathrm{B}(x, d)|$ ▷*Sample next walk position x^l inside* B
  $W \leftarrow |\mathrm{B}(x, d)| \cdot G^{\overline{\sigma}, \mathrm{B}}(x^c, x^l) \cdot (\overline{\sigma} - \sigma'(x^l)) \cdot W$ ▷*Update path throughput*
  $\widehat{I}_f \leftarrow \widehat{I}_f + f(x^l) \cdot W / \sqrt{\kappa(x^l)} \cdot (\overline{\sigma} - \sigma'(x^l))$ ▷*Accumulate source contribution*
  $x^c \leftarrow x^l$ ▷*Update current walk position*
return $\frac{1}{\sqrt{\kappa(x)}} \left(\sqrt{\kappa(z)} \cdot \text{NextFlightWoS}(z, \varepsilon) \cdot \widehat{T} + \widehat{I}_f \right)$ ▷*Repeat from next walk position z*

be used to evaluate the following integral expression for $\nabla_x u(x)$ in a ball $\mathrm{B}(x, R)$:

$$\nabla_x u(x) = \frac{1}{\sqrt{\kappa(x)}} \left(\int_{\partial \mathrm{B}(c,R)} \nabla_x P^{\overline{\sigma}, \mathrm{B}}(x, z)\ \sqrt{\kappa(z)}\ u(z)\, \mathrm{d}z\ +\ \int_{\mathrm{B}(c,R)} \nabla_x G^{\overline{\sigma}, \mathrm{B}}(x, y)\ f'(y, \sqrt{\kappa}\ u)\, \mathrm{d}y \right) - \frac{u(x)}{2\kappa(x)} \nabla_x \kappa(x). \tag{5.14}$$

Similar to Sec. 3.2.5, the value of $\nabla_x u(x)$ only needs to be estimated for the first ball in any walk—the solution values u, on $\partial\mathrm{B}$ and inside B, that the gradient estimate depends on can be computed recursively using the delta tracking or next flight estimator. The parameters $\overline{\sigma}$, p^{B}, $p^{\partial \mathrm{B}}$, $\mathbb{P}^{\mathrm{B}}$ and $\mathbb{P}^{\partial \mathrm{B}}$ remain unchanged from Sec. 5.2 with either estimator.

5.3.2 Reflecting Boundary Conditions

It is straightforward to show how reflecting boundary conditions change under a Girsanov transformation. For instance, assume the normal derivative $\partial u / \partial n = h$ is prescribed in Eq. 5.1 on a Neumann boundary $\partial\Omega_{\mathrm{N}}$. Moreover, from Eq. 5.3, we know that the function $U(x) = \sqrt{\kappa(x)} u(x)$ solves a variable screened Poisson equation. Computing its normal derivative yields

$$\frac{\partial U(x)}{\partial n_x} + \mu'(x) U(x) = h'(x), \tag{5.15}$$

$$\text{where} \quad \mu'(x) := -\frac{1}{2} \frac{\partial \ln(\kappa(x))}{\partial n_x} \quad \text{and} \quad h'(x) := \sqrt{\kappa(x)}\, h(x).$$

A delta tracking or next flight variant of the walk on stars estimator from Ch. 4 could in principle deal with the Robin boundary conditions in Eq. 5.15, and simultaneously resolve spatial variability in the PDE coefficients (repeating the derivation in Sec. 5.1 gives an integral expression similar to Eq. 5.9). However, the Robin coefficient μ' in this scenario is not a strictly positive function, which is a key assumption made with the WoSt estimator in Sec. 4.3. We leave a general treatment of variable-coefficient PDEs with reflecting boundary conditions to future work.

Chapter 6

Solver Implementation & Tuning

Similar to a scene in a renderer, a PDE is encoded by a description of the domain boundary $\partial\Omega$, boundary conditions g, h, ℓ (Eq. 2.1), source term f, and PDE coefficients κ, $\vec{\omega}$, σ (Sec. 2.1.1 & App. E). We implement inputs as arbitrary callback routines that return a value for any query point $x \in \Omega$. Unlike conventional solvers such as FEM and finite differences, problem inputs need not be discretized or approximated in a finite-dimensional basis. The gradient and Laplacian of the coefficients $\kappa(x)$ and $\vec{\omega}(x)$ (Eq. 5.3 & E.8) can be evaluated via any standard technique (*e.g.*, automatic differentiation), while the bounding parameter $\overline{\sigma} \coloneqq \max(\sigma'(x)) - \min(\sigma'(x))$ is computed as in volume rendering [188], *e.g.*, by regular, random or progressive sampling [169].

In this chapter, we describe how to efficiently implement the geometric queries on $\partial\Omega$ needed by the walk on spheres and walk on stars algorithms (Sec. 6.1 & 6.2). In Sec. 6.3, we detail the impact of the ε-shell parameter on the estimated PDE solution with Dirichlet, Neumann or Robin conditions, and discuss convergence with increasing number of walks. We also provide a reference implementation of these PDE estimators in an open-source solver called Zombie [222] (in homage to "random walks").

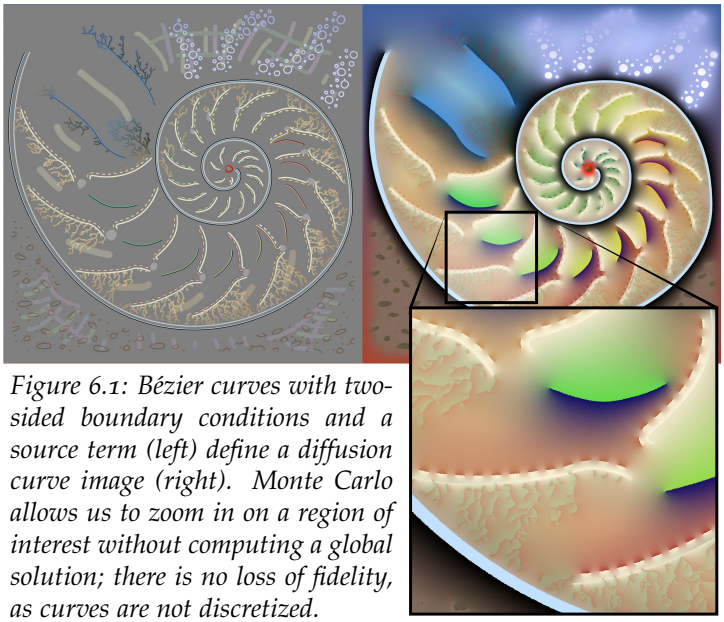


Figure 6.1: Bézier curves with two-sided boundary conditions and a source term (left) define a diffusion curve image (right). Monte Carlo allows us to zoom in on a region of interest without computing a global solution; there is no loss of fidelity, as curves are not discretized.

All estimators can evaluate the solution and its gradient at arbitrary points in the domain without a global solve. Since values are estimated independently at each point, the implementation is *embarrassingly parallel* and output-sensitive, *i.e.*, the solve can be performed at any resolution locally (Fig. 1.9, 6.1, 8.4 & 8.6), rather than always on the entirety of a background grid. Moreover, as in rendering, we can progressively increase the number of walks per point to generate a "preview" that can be subsequently refined (Fig. 1.2, 1.9 & 4.2). This approach provides a fast iteration cycle (Fig. 6.2 shows an example where the geometry and boundary conditions are edited interactively), and is especially valuable when working with massive models (Sec. 8.1).

6.1 Closest Point Queries For Walk On Spheres

To find an empty ball $\mathrm{B}(x, R)$ around a given point $x \in \Omega$ for the WoS algorithms in Sec. 3.2 and Ch. 5, we need only determine the distance R to the closest point $\overline{x} \in \partial\Omega$, or a conservative underestimate of this distance. As detailed here, such distances are easily computed for a wide

ALGORITHM 6: $\textsc{ClosestPt}(T \leftarrow \textsc{Bvh}(\partial\Omega_{\mathrm{D}}),\ x,\ \overline{x} \leftarrow \textsc{null},\ d \leftarrow \infty,\ d_T^{\min} \leftarrow 0)$

Input: Bounding volume hierarchy T, query point $x \in \mathbb{R}^3$, closest point $\overline{x}$ to be determined, current estimate d for distance, and minimum distance $d_T^{\min}$ to T's axis aligned bounding box from x ($d_T^{\min} = 0$ when $x \in T.\mathrm{aabb}$).

Output: Distance to $\partial\Omega_{\mathrm{D}}$, and closest point on $\partial\Omega_{\mathrm{D}}$.

if $d_T^{\min} > d$ **then return** $d, \overline{x}$ ▷*Ignore current BVH node if it is further than d*
if $T.\mathrm{isLeaf}$ **then**
  for t **in** $T.\mathrm{triangles}$ **do**
    $d_t, \overline{x}_t \leftarrow \textsc{ClosestPt}(t,\ x)$ ▷*Compute closest pt $\overline{x}_t$ on triangle t from x [62, Sec. 5.1.5]*
    if $d_t < d$ **then** $d, \overline{x} \leftarrow d_t, \overline{x}_t$ ▷*Update d and $\overline{x}$ if $\overline{x}_t$ is closer than d*
else
  $\mathrm{visitLeft}, d_{\mathrm{left}}^{\min}, d_{\mathrm{left}}^{\max} \leftarrow \textsc{Intersect}(T.\mathrm{left}, x, d)$ ▷*Check if left* aabb *intersects ball* $\mathrm{B}(x, d)$
  if visitLeft **then** $d \leftarrow \min(d,\ d_{\mathrm{left}}^{\max})$ ▷*d cannot be greater than max distance to left* aabb
  $\mathrm{visitRight}, d_{\mathrm{right}}^{\min}, d_{\mathrm{right}}^{\max} \leftarrow \textsc{Intersect}(T.\mathrm{right}, x, d)$ ▷*Check if right* aabb *intersects ball* $\mathrm{B}(x, d)$
  if visitRight **then** $d \leftarrow \min(d,\ d_{\mathrm{right}}^{\max})$ ▷*d cannot be greater than max distance to right* aabb
  if visitLeft **and** visitRight **then**
    if $d_{\mathrm{left}}^{\min} < d_{\mathrm{right}}^{\min}$ **then** ▷*Visit closer node first*
      $d, \overline{x} \leftarrow \textsc{ClosestPt}(T.\mathrm{left},\ x,\ \overline{x},\ d,\ d_{\mathrm{left}}^{\min})$
      $d, \overline{x} \leftarrow \textsc{ClosestPt}(T.\mathrm{right},\ x,\ \overline{x},\ d,\ d_{\mathrm{right}}^{\min})$
    else
      $d, \overline{x} \leftarrow \textsc{ClosestPt}(T.\mathrm{right},\ x,\ \overline{x},\ d,\ d_{\mathrm{right}}^{\min})$
      $d, \overline{x} \leftarrow \textsc{ClosestPt}(T.\mathrm{left},\ x,\ \overline{x},\ d,\ d_{\mathrm{left}}^{\min})$
  else if visitLeft **then** $d, \overline{x} \leftarrow \textsc{ClosestPt}(T.\mathrm{left},\ x,\ \overline{x},\ d,\ d_{\mathrm{left}}^{\min})$ ▷*Visit left node*
  else if visitRight **then** $d, \overline{x} \leftarrow \textsc{ClosestPt}(T.\mathrm{right},\ x,\ \overline{x},\ d,\ d_{\mathrm{right}}^{\min})$ ▷*Visit right node*
return $d, \overline{x}$

variety of shapes; multiple shapes can be combined by taking the minimum over all per-shape distances, or more generally, by applying Boolean operations (discussed below).

Closest point queries (CPQs) can be accelerated via a spatial hierarchy [203, Ch. 4]. Relative to ray tracing [197, 259], there has been little work on high-performance CPQs [138, 232, 273, 287], though recent GPUs provide opportunities for massive acceleration [260]. We employ an axis-aligned boundary value hierarchy built using the *surface area heuristic (SAH)* [261], via the FCPW library [220]—unlike ray tracing, oriented bounding volumes [62, 143] can provide more effective culling, and hence further speedups. Alg. 6 provides pseudocode for closest point traversal with a binary BVH containing triangle primitives. The queries described in Sec. 6.2 for WoSt will follow the same basic traversal strategy (Alg. 9), but will use slightly modified criteria to determine whether a node in the hierarchy should be visited (Alg. 10).

Polygon Soup. Real-world geometry is often given as a list of polygons without explicit connectivity, and which satisfies no special conditions (manifold, orientable, *etc.*). We can solve PDEs with Dirichlet conditions directly on such "polygon soups" by taking the closest point among all polygons. Rather than attempt to fix cracks, holes, and self-intersections [8, 233], we simply solve an exterior problem (Sec. 3.2.4), as shown in Fig. 1.7 (*right*). Note that in contrast to generalized winding numbers [114], the input need not meet any special conditions on orientation.

Parametric Curves & Surfaces. We use the method of Chen et al. [37] to compute closest points on 2D cubic Bézier curves. Such a representation is attractive for illustration tools (*e.g.*, *Illustrator* or *Inkscape*), as it avoids mesh generation and quantization error (consider Fig. 6.1 & 8.7). Closest points can also be computed directly for NURBS and subdivision surfaces [58, 158, 249].

Implicit Surfaces. Many shapes are concisely described by the zero level set of a function $\phi : \mathbb{R}^N \to \mathbb{R}$. When ϕ is a signed distance function, the size of the largest ball around x is simply $|\phi(x)|$. More generally, conservative estimates of the distance to an implicit surface can be obtained by bounding the gradient $|\nabla\phi|$, which in turn gives a Lipschitz constant for ϕ [97, Theorem 1]. A rich variety of shapes can be described this way [97, Table 1]; Fig. 1.7 (*left*) shows a smooth blend between two tori.

Booleans. Boolean operations are used to concisely encode complex models, via *constructive solid geometry (CSG)* [209]. Tremendous effort has been put into developing robust *mesh booleans* [19, 200, 281], but they generally remain expensive and error prone, cannot be mixed with other geometric representations, and still require meshing of the domain interior. In contrast, ray tracing can evaluate booleans via simple arithmetic on intersection distances [212]. We can likewise combine closest point distances to solve PDEs with Dirichlet boundary conditions directly on boolean arrangements (Fig. 1.7 & 6.2). We refer to Hart [97, Table 1] for operations on distances needed for both hard booleans and soft "blends".

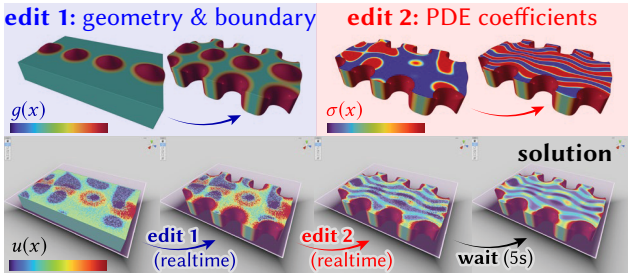


Figure 6.2: Our Monte Carlo approach operates directly on the original scene representation (here, signed distance fields composed via CSG operations), and provides instant feedback after updates to the scene geometry and boundary conditions (edit 1) or PDE coefficients (edit 2).

6.2 Accelerated Geometric Queries For Walk On Stars

In general, WoSt works with any boundary representation that supports the following queries:

1. closest point queries on $\partial\Omega_D$,
2. closest silhouette point queries on $\partial\Omega_N$,
3. star region radius queries on $\partial\Omega_R$,
4. ray intersection queries against $\partial\Omega_N$ and $\partial\Omega_R$, and
5. point sampling queries on $\partial\Omega_N$ and $\partial\Omega_R$.

Since none of these queries require the boundary to have a well-defined inside and outside, $\partial\Omega$ need not be watertight, and can have cracks, holes, or self-intersections—see in particular App. B for a discussion of open domains and double-sided boundary conditions.

In principle, these queries could be evaluated for, say, spline patches or implicit surfaces (as in Sec. 6.1); we focus exclusively on triangle meshes here. In particular, ray intersections are standard in computer graphics, and can be accelerated using a BVH (just like CPQs). Sec. 6.2.1 describes the closest silhouette point query for Neumann boundaries. The star region radius

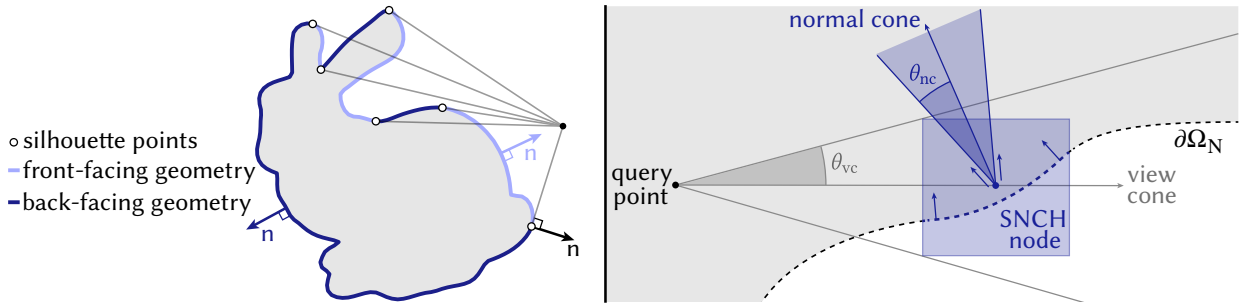


Figure 6.3: Left: An optimized procedure for finding silhouettes should avoid visiting finely-tessellated geometry that is entirely front- or back-facing relative to the query point. Right: A SNCH tests for a pair of mutually orthogonal directions in a view cone and a node's normal cone to determine whether the node contains a silhouette edge on the Neumann boundary. The geometry inside the node can be skipped if no such pair of directions is found.

query for Robin boundaries, detailed in Sec. 6.2.2, functions as a CSPQ when the Robin coefficient $\mu = 0$, and as a CPQ when $\mu = \infty$. For intermediate values of μ, this query has additional aspects unique to Robin conditions, which we highlight in green in Alg. 8 & 10. The point sampling query (Alg. 11) is designed to sample known Neumann data h and Robin data ℓ on reflecting boundaries, and is discussed in Sec. 6.2.4.

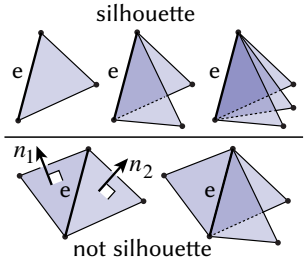
All queries use a spatial hierarchy to visit only a small fraction of the triangles on the boundary of the input domain; our basic approach is to build a *spatialized normal cone hierarchy (SNCH)* [119] by adding normal information to the BVH already used by WoS. In particular, we use a standard BVH to perform CPQs on the Dirichlet boundary, and a separate SNCH for queries on the Neumann and Robin boundaries, using normal information only for queries 2 & 3. In practice, all queries needed to implement WoSt on triangle meshes are supported by the open-source FCPW library [220]; see also App. F for pseudocode.

6.2.1 Closest Silhouette Point Query For Neumann Boundaries

The silhouette of a triangle mesh, relative to a given direction v, occurs along a set of edges e that satisfy a local silhouette condition. In particular, e is a silhouette edge if for each distinct pair of triangles containing e,

$$(v \cdot n_1) \cdot (v \cdot n_2) \leq 0, \tag{6.1}$$

where n_1, n_2 are consistently oriented normals (see inset). Note in particular that every boundary edge is a silhouette edge. A naïve strategy for finding the closest silhouette edge is to use a BVH to locate the closest point to x on all edges, skipping edges not contained in the silhouette. However, this strategy is highly inefficient when BVH nodes contain large, finely-tessellated regions that are all front- or back-facing (Fig. 6.3, *left*): here each edge is examined (and rejected) exhaustively, whereas ideally the whole node should simply be culled.



Spatialized Normal Cone Hierarchy

To improve scaling, we hence augment our BVH with information about the orientation of the geometry inside each node. In particular, we adopt the spatialized normal cone hierarchy of Johnson and Cohen [119]. Each node of a SNHC stores not only an axis-aligned bounding box (AABB), but also a *normal cone*. The cone axis is the average normal of all triangles in the node,

and the cone half angle θ is the maximum angle between the axis and any triangle normal (Fig. 6.3, *right*). Normal cones can be assembled during BVH construction. We use the surface area heuristic; performance could possibly be further improved via the *surface area orientation heuristic (SAOH)* of Conty Estevez and Kulla [39, Sec. 4.4], which clusters primitives according to both proximity and alignment.

Closest Silhouette Point Traversal

To perform a silhouette query on $\partial\Omega_{\mathrm{N}}$, we traverse the SNCH in depth-first order (Alg. 9). For each node in this traversal we build a *view cone* rooted at x. The cone's axis points toward the center of the node, and its half-angle tightly bounds the AABB (Fig. 6.3, *right*); we then check if the view cone and the node's normal cone contain a pair of mutually orthogonal directions. If this test fails, all triangles in the node must be front- or back-facing relative to the query point, and the node can be skipped. In the context of WoSt, an upper bound on the size of a star-shaped region $\mathrm{St}(x, R)$ is given by the distance $d_{\mathrm{Dirichlet}}$ from x to the Dirichlet boundary (Sec. 4.2.2). To further improve query efficiency we can hence restrict the search to the radius $R^{\max} = d_{\mathrm{Dirichlet}}$.

6.2.2 Star Region Radius Query For Robin Boundaries

The naïve approach for computing the radius of a star-shaped region on $\partial\Omega_{\mathrm{R}}$ is to first perform a CSPQ relative to a query point x, and then to loop over all triangles within the radius returned by the CSPQ, estimate the upper-bound on the radius for each triangle t (Eq. 4.11), and take the minimum of these bounds. Before we describe how to accelerate this computation using an SNCH, we observe that the radius bound does not have to be approximated numerically for any t. We can instead compute a tight bound using the maximum coefficient value $\mu^{\max} := \max(\mu(z))$[1] over all points $z \in t$, and a distance h from x to the plane t lies on. In particular, letting $r = {}^{h}/_{\cos\theta}$ in Eq. 4.11, we have:

$$R \leq \frac{\mu^{\max}\, h^2}{\mu^{\max}\, h\, \cos\theta - \cos^3\theta} \quad \text{when } \cos\theta \leq \sqrt{\mu^{\max}\, h}. \tag{6.2}$$

We now minimize this equation by taking its derivative with respect to $\cos\theta$, and setting it to zero. This gives an analytical expression $\sqrt{\mu^{\max} h / 3}$ for the cosine. We clamp this expression between the minimum and maximum cosine values achieved at the closest and farthest points on t (respectively), and plug it back into Eq. 6.2 to compute the radius bound for t. Alg. 8 provides pseudocode.

6.2.3 Accelerating Star Region Queries

Not every triangle in a mesh needs to be visited to compute the radius of a star-shaped region. In fact, as we search for the minimum upper-bound on the radius over all triangles, we can skip over large parts of $\partial\Omega_{\mathrm{R}}$ where the geometry is entirely front- or back-facing relative to the query point x (akin to a CSPQ). The only alteration we make to the SNCH from Sec. 6.2.1 is to also store minimum and maximum Robin coefficient values over all triangles in a node.

[1] Using $\mu^{\max}$ to compute the radius bound for a triangle does not alter the original problem description—the WoSt estimator will still treat $\mu(z)$ as spatially-varying over the triangle when computing reflectance in Alg. 3, *line* 1.

Query Traversal. Similar to Eq. 6.2, we compute *conservative* radius bounds for the nodes we visit during traversal, using the spatial, angular, and coefficient information available in a SNCH node (Alg. 10, *lines* 7-10). We also build a view cone rooted at x to compute the bounds, via Eq. 4.11. We then decide whether a node with only front or back-facing triangles can be skipped, by checking if our minimum bound for the node is larger than the current estimate of the radius for the star-shaped region (Alg. 9, *line* 1). We also use our maximum bound for the node to shrink the radius estimate (Alg. 9, *lines* 7 & 9). If instead the node contains a geometric silhouette (Alg. 10, *line* 4), we must traverse the node just as with a CSPQ, since Eq. 4.11 only applies to star-shaped regions which can be no larger in size than the distance returned by a CSPQ.

6.2.4 Point Sampling Query

Recall that for problems with nonzero Neumann and Robin conditions, we sample points on $\partial\Omega_N$ and $\partial\Omega_R$ respectively (Sec. 4.2.3). To increase the likelihood that these points lie on $\partial\mathrm{St}_N$ and $\partial\mathrm{St}_R$, we adopt a *hierarchical importance sampling* strategy used in rendering to accelerate next-event estimation [39]. In particular, during each step of a BVH traversal, we select only a single *random* child whose AABB intersects $\mathrm{St}(x, R)$. To give preference to nodes closer to the query point x, we sample according to the free-space Green's function $G^{\mathbb{R}^N}$ (Alg. 12, *lines* 11-14), rather than the Green's function for a ball (which becomes negative outside $\mathrm{St}(x, R)$). Once we reach a leaf node, we uniformly sample a point from the leaf triangles with respect to surface area (Alg. 11, *line* 3). This point is not guaranteed to lie on $\partial\mathrm{St}_N$ or $\partial\mathrm{St}_R$, but is much more likely to do so compared to uniformly sampling all of $\partial\Omega_N$ or $\partial\Omega_R$ (respectively). Conty Estevez and Kulla [39, Sec. 5.4] describe further improvements to this traversal strategy.

6.3 Epsilon Parameter & Convergence Of Estimators

Our solvers use a single parameter to control the thickness of the epsilon shell $\partial\Omega^\varepsilon$, irrespective of the type of boundary condition prescribed on $\partial\Omega$. This parameter trades bias in the solution estimate with the number of steps in a random walk, and in general requires little-to-no hand-tuning, as bias drops predictably with decreasing values of ε.

Dirichlet Boundary. Early stopping extends boundary values into an epsilon-neighborhood $\partial\Omega_D^\varepsilon$, and the solution in turn exhibits a small bias toward these values. Dirichlet boundary conditions are hence still enforced with the given data, but the location of enforcement may be off by a tiny distance ε. Though in principle bias can be completely eliminated via a *Green's function first passage* approach [81, 159], a pragmatic solution is to simply use a small value of ε.

In practice, an accurate solution is obtained for fairly large ε values, even in the presence of tiny features (Fig. 6.4). Importantly, arbitrarily small geometric features will *always* appear in the solution, since there will always be evaluation points x in their Voronoi region, independent of ε. Moreover, such features will still have a global effect on the solution, since a random walk has a nonzero probability of reaching any boundary component of finite size. The only potential problem is if spacing *between* features is smaller than ε, which is easily avoided by (universally or adaptively) using a small ε. Note that shrinking ε by a few orders of magnitude does not significantly increase cost due to the exponential shrinking of balls [22]. In contrast, mesh-based solvers can eliminate small features entirely, and their time/memory cost often blows up dramatically with smaller tolerances ε (Fig. 1.3, 8.1, 8.4 & 8.10).

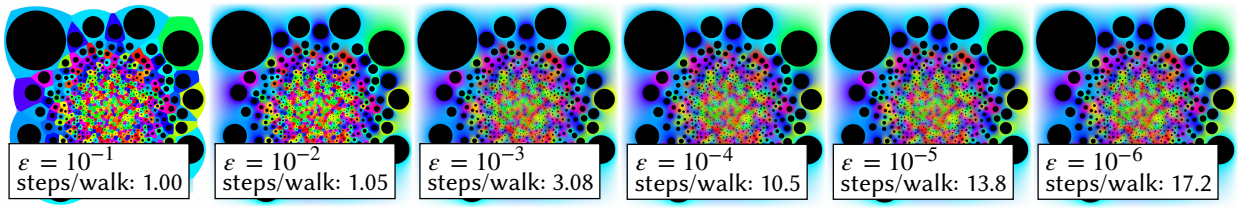


Figure 6.4: Tiny features are preserved for any value of ε. For very large values of ε, WoS algorithms jump to the closest point, producing a Voronoi-like solution (left). Decreasing ε quickly eliminates any bias. Since small ε values do not significantly increase the average number of steps per walk, it is generally unnecessary to hand-tune this parameter. (Note that if the box above represents a domain 1 meter in width, then 10^{-4} is about the width of a human hair; small bacteria are on the order of 10^{-6}.)

Neumann And Robin Boundaries. Enforcing a minimum radius on the size of a star-shaped region (Sec. 4.2.2) also incurs bias near concave parts of $\partial\Omega_N$ and $\partial\Omega_R$. This bias manifests as a global darkening of the estimated solution as ε is increased. Fig. 6.5 examines the effect of this parameter on a Neumann problem—compared to Dirichlet conditions, here the performance-bias tradeoff is more sensitive to values of ε. However as before, the relative rate at which bias grows with increasing ε values is outweighed by the performance improvement from walks taking larger steps on $\partial\Omega_N$, and terminating faster on $\partial\Omega_D$. Adaptively picking ε based on local boundary curvature should yield better performance and lower bias.

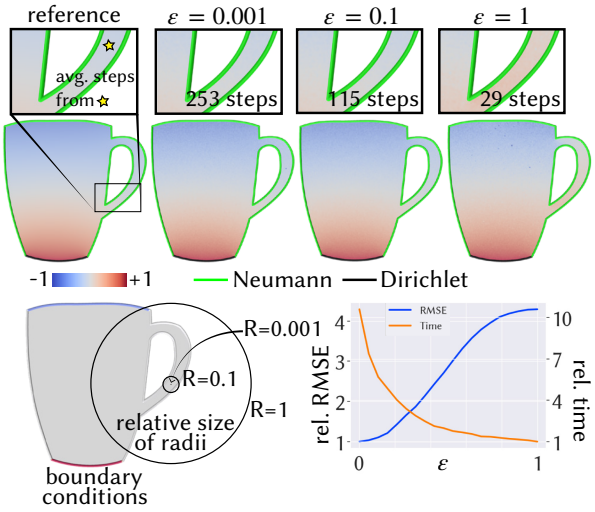


Figure 6.5: WoSt uses an ε-shell to also ensure that random walks make progress near concave parts of a reflecting boundary. Walks generally converge faster with larger values of ε, with run-time improvements outweighing the relative increase in bias.

Fig. 6.6 examines the impact of ε on a Robin problem with both more absorbing ($\mu > 1$) and more reflecting ($\mu < 1$) boundary conditions—in the limit, we recover the affect of ε on pure Dirichlet and Neumann problems respectively. In all other experiments, we scale models to fit in a unit sphere, and use $\varepsilon = 0.001$ with all boundary conditions.

Convergence. All PDE estimators from Ch. 3–5 exhibit the expected $O(1/\sqrt{N})$ Monte Carlo rate of convergence with respect to the number of walks N (Sec. 3.1), suggesting that any bias from the sole ε-shell parameter has little impact on overall accuracy; see Fig. 5.4, 6.6, 8.2, 8.14, 8.16 & 8.17. In general, variance tends to be larger (but still predictable) with recursive estimation of higher-dimensional integrals, *i.e.*, it is larger in regions where walks are longer—examples include Neumann dominated problems (Fig. 8.17, *bottom center*), and PDEs with high spatial variability in their coefficients (Fig. 7.20). We describe variance reduction strategies for these estimators in the next chapter, which do not increase the rate of convergence, but they do lower the magnitude of the error in the estimated results (*e.g.*, Fig. 7.4, 7.17 & 7.20).

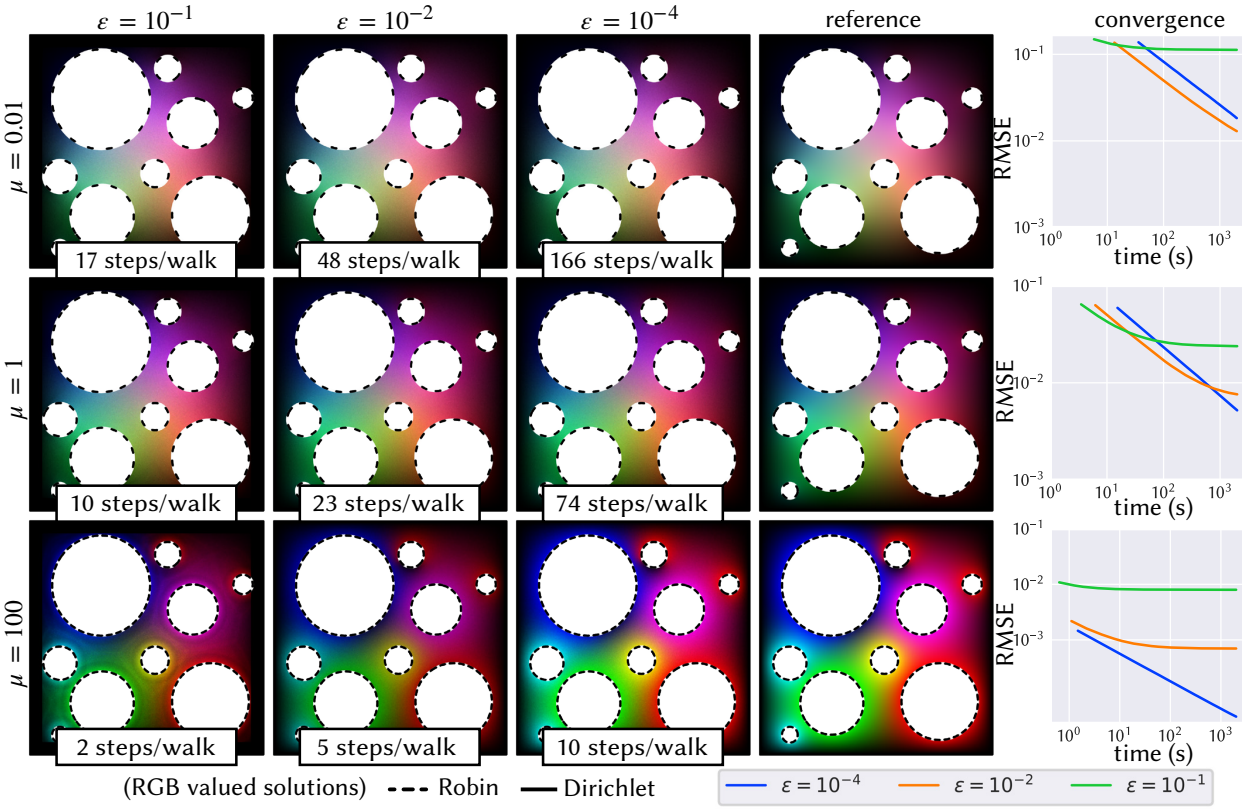


Figure 6.6: Top two rows: For more reflecting Robin boundaries with smaller coefficients μ, bias manifests as a global darkening in the solution estimate for large ε. Bottom row: For more absorbing Robin boundaries with larger coefficients μ, a large ε-shell extends prescribed boundary values further into the domain interior, similar to the bias observed with pure Dirichlet problems.

Chapter 7

Variance Reduction

Like any other Monte Carlo method, we can reduce the noise in our PDE estimators by simply taking more samples, *i.e.*, by performing more random walks from an evaluation point. However, in many application scenarios, the solution must be estimated within a fixed time constraint, or using a given sample budget. Therefore, it is often not enough to just design Monte Carlo estimators—we must also look for ways to make them more *efficient* (Eq. 3.5).

In this chapter, we develop several variance reduction strategies for the estimators developed in Ch. 3–5, by leveraging the special structure of integral equations corresponding to elliptic PDEs. Some of these strategies are inspired by analogous methods in Monte Carlo rendering as the integrals share similarities (*e.g.*, Sec. 7.1, 7.4, 7.5 & 7.6), while others are specifically designed for elliptic equations (*e.g.*, Sec 7.2, 7.3 & 7.7). Most strategies are complementary to each other (and can hence be used in conjunction), as they target different facets of our estimators. Though we achieve noticeable improvement in estimation quality here, there remains significant scope for further variance reduction—as will be discussed in Ch. 9.

7.1 Importance Sampling Of Source Terms

Recall that the integral formula for a Poisson equation involves a term $\int_{B(x,R)} G^{\mathrm{B}}(x,y) f(y)\,\mathrm{d}y$, where G^{B} is the harmonic Green's function on $\mathrm{B}(x,R)$. One way to importance sample this term is to simply draw y from the distribution $p^{\mathrm{B}} := G^{\mathrm{B}} / \int_{\mathrm{B}(x,R)} G(x,y)\,\mathrm{d}y$, as previously discussed in Sec. 3.2.1. Such samples can be generated via standard techniques, *e.g.*, rejection sampling (App. A.1.2 & A.2.2); in 3D we use *Ulrich's polar method* [54, Section 9.4]. In principle, one could extend this strategy to PDEs where the Green's function can only be tabulated numerically, akin to importance sampling of measured BRDFs [145].

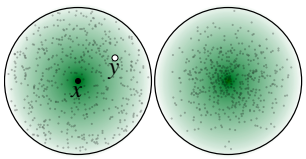


Figure 7.1: Uniform (left) vs importance sampling (right) of the Green's function.

We can also sample the source term f. An important case is a *point source* $f_z := c\delta_z$, where $c \in \mathbb{R}$ is a constant, and δ_z is the Dirac delta centered at a point $z \in \Omega$. If a single δ_z is inside the current ball $\mathrm{B}(x,R)$, we can use an importance density $p^{\mathrm{B}} = \delta_z$, yielding the estimator

$$\frac{G^{\mathrm{B}}(x,y)\ f(y)}{p^{\mathrm{B}}(y)} = \frac{G^{\mathrm{B}}(x,y)\ c\delta_z(y)}{\delta_z(y)} = cG^{\mathrm{B}}(x,z), \tag{7.1}$$

i.e., just use a single sample at $y = z$. Similarly, consider a *curve source* $f_\gamma := c\delta_\gamma$, where δ_γ is the (1-Hausdorff) measure of a curve $\gamma \subset \Omega$, and c is a function along γ. When γ intersects B, we can sample points $y_1, \ldots, y_M$ uniformly from $\gamma_{\mathrm{B}} := \gamma \cap \mathrm{B}$, and use the estimator

$$|\gamma_{\mathrm{B}}| \frac{1}{M} \sum_{i=1}^{M} c(y^i) G(x,y^i). \tag{7.2}$$

Alternatively, one can just uniformly sample the whole curve γ, and drop the contribution of

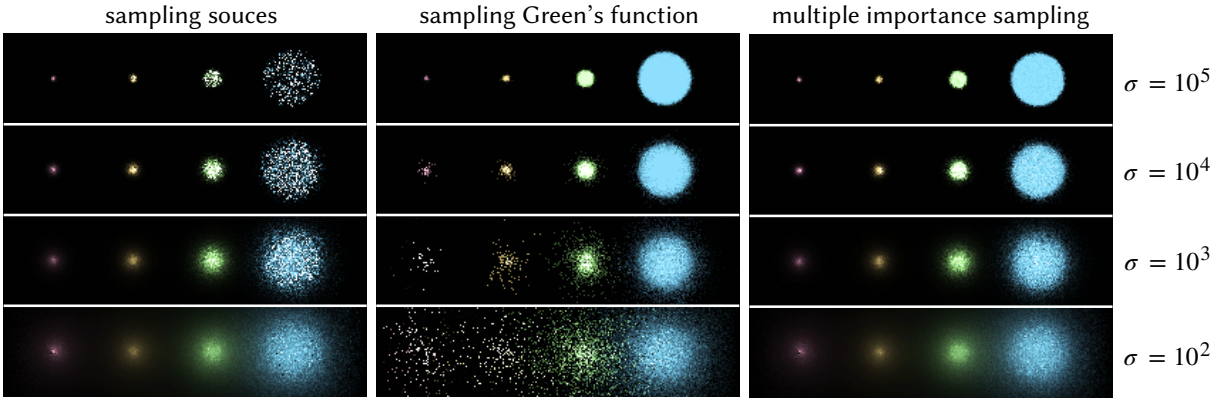


Figure 7.3: Here we use multiple importance sampling [252] of the Green's function $G^{\sigma,\mathrm{B}}$ and source term f to robustly sample screened Poisson equations for different coefficient values σ with area sources of varying size.

points outside B. This strategy is easily generalized to any m-dimensional subset; Fig. 7.2 shows an example.

The importance sampling strategies described above, with G^{B} and f as the sampling densities, can also be combined using *multiple importance sampling* [252], as in Fig. 7.3. We note that Monte Carlo is often better suited than quadrature methods for integrands with singularities—importance sampling can be applied to handle such integrands effectively, even in situations where there is no analytic transformation to remove the singularity.

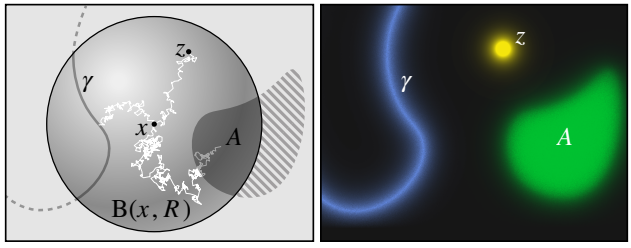
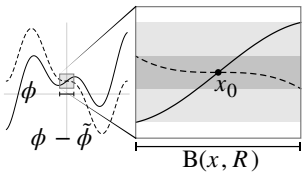


Figure 7.2: Source terms can be importance sampled by randomly picking points on the source where it is non-zero, restricted to the current ball in a random walk. Without importance sampling, only the area source f_A would appear on the right; the point source f_z and curve source f_γ would never get sampled.

7.2 Control Variates



Suppose we let $\tilde{\phi}$ be a low-order Taylor approximation of ϕ in Eq. 3.7 around some point x_0. Then the function $\phi - \tilde{\phi}$ will look "flat" in a small neighborhood of x_0, and estimating it will yield low variance. This control variate is useful for WoS estimators, which seek to integrate a function over a typically small sphere or ball. It also applies to WoSt if we use an empty ball, in place of a star-shaped region, for just the first step of a random walk. Though we do not know the terms of the Taylor series *a priori*, we can use running derivative estimates to get an increasingly good guess. In fact, when both the solution and gradient strategies decribed below are used in conjunction, they reinforce each other: the variance of the solution estimator is reduced by the gradient estimate, and vice versa. In practice, we use these two strategies for all PDEs.

Control Variate For Solution

For a PDE with solution u, let $\widehat{\nabla u_k}(x_0)$ be the running estimate of the gradient for the first k walks at x_0 (computed as in Sec. 3.2.5). Since the linear term $\nabla u(x_0) \cdot (x - x_0)$ in the Taylor series for $u(x_0)$ integrates to zero over any ball around x_0 (*i.e.*, $c = 0$ in Eq. 3.7), we can replace an

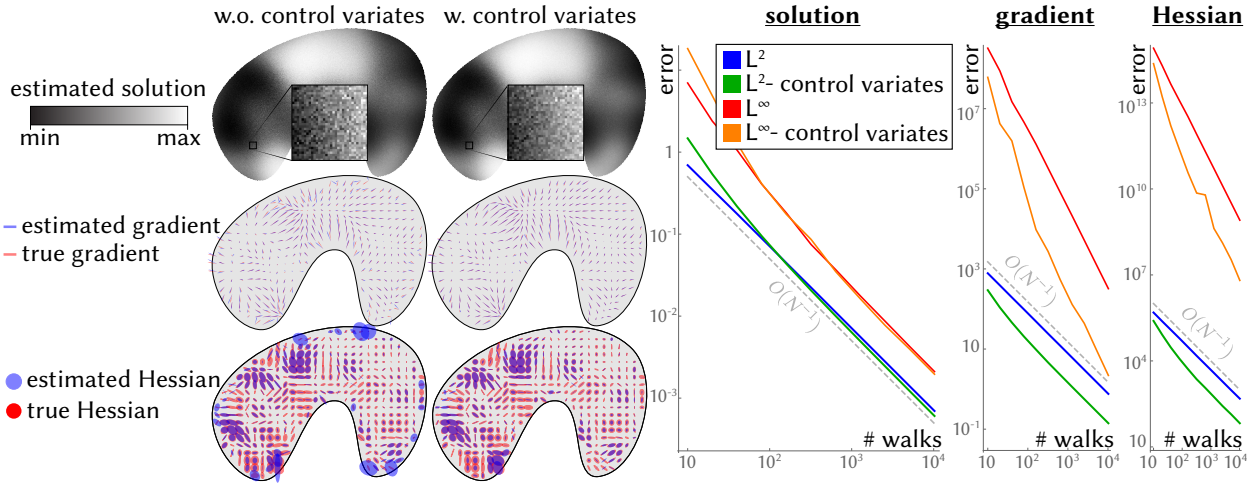


Figure 7.4: Control variates provide modest variance reduction for the solution, but become more important for higher derivatives of the solution. Here we plot the Hessian's principal values and axes as ellipses; notice that variance is higher near the boundary, and axes are harder to estimate in regions where the Hessian has small magnitude.

existing WoS estimator $\widehat{u}$ with

$$\frac{1}{N}\sum_{i=1}^{N}\widehat{u}(x_1^i) - \widehat{\nabla u}_{i-1}(x_0)\cdot(x_1^i - x_0), \quad x_1^i \sim \mathscr{U}(\partial \mathrm{B}(x_0, R)), \tag{7.3}$$

where x_1^i denotes the first step in the ith walk. Note also that the estimate $\widehat{\nabla u}_{i-1}$ is statistically independent of the ith walk. In practice even this simple strategy can help reduce variance—see for example Fig. 7.4.

Control Variate For Gradient

Variance reduction for derivatives is especially important, since differentiation amplifies high frequencies. Our control variate strategy for the gradient is complementary to the one for the solution: let $\widehat{u}_k(x_0)$ be the running estimate of the solution $u(x_0)$ for the first k walks. Then we can replace the gradient estimate for the boundary term (Eq. 3.18) with

$$\frac{d}{R}\frac{1}{N}\sum_{i=1}^{N}(\widehat{u}(x_1^i) - \overline{u}_{i-1}(x_0))\cdot n_{x_1^i}. \tag{7.4}$$

Since by symmetry the normal $n_{x_1^i}$ integrates to zero over a sphere, the expected value of the estimator is unchanged (*i.e.*, $c = 0$), but the variance is typically lower as the control variate approaches the true value of $u(x_0)$. Likewise, if $f_k := \frac{1}{k}\sum_{i=1}^{k} f(y^i)$ is the running average of source values sampled from the initial ball, then we can subtract this value from $f(y)$ in our estimator for the initial source term in Eq. 3.14.

This strategy parallels methods used in reinforcement learning [245, Section 13.4], and is related to techniques used on discrete grids [183]. We refer to Rioux-Lavoie et al. [211, Sec. 4] for an antithetic variate strategy (Sec. 3.1.1) for the gradient estimator, which can be used alongside our control variate to reduce variance even further. Both strategies are particularly effective when estimating the gradient in the vicinity of the domain boundary, where the ball size is smaller.

Control Variate For Hessian

Control variates can also be applied to higher-order derivatives. For example, in the ith term of our Hessian estimator for the boundary integral (Eq. 3.23), we can replace $\widehat{u}(x_1^i)$ with

$$\widehat{u}(x_1^i) - \widehat{u}_{i-1}(x_0) - \widehat{\nabla u}_{i-1}(x_0) \cdot (x_1^i - x_0). \tag{7.5}$$

Alanko and Avellaneda [3] discuss a similar approach for grids. Note that both running sums have already been computed for the source and gradient control variates—these could now be further improved via the Hessian estimate. Fig. 7.4 shows the effect on variance, which is about 7x lower than the baseline estimator from Eq. 3.23.

7.3 Tikhonov Regularization

The solution to a Poisson equation with pure Neumann conditions is determined only up to an additive constant. When we solve such a PDE with WoSt, we observe that high frequency details in the PDE solution are often resolved by the first few steps of a random walk, while the contribution from later steps is closer to constant; see Fig. 3.6. As mentioned in Sec. 4.2, we use Tikhonov regularization (Sec. 3.2.6) to more effectively handle such problems—Fig. 7.5 shows this approach provides estimates without much noise or bias even with substantial regularization, while ensuring walk length is not unbounded.In general, the number of steps needed to resolve the solution is problem-dependent, and more steps are typically needed when the solution has low-frequency global features.

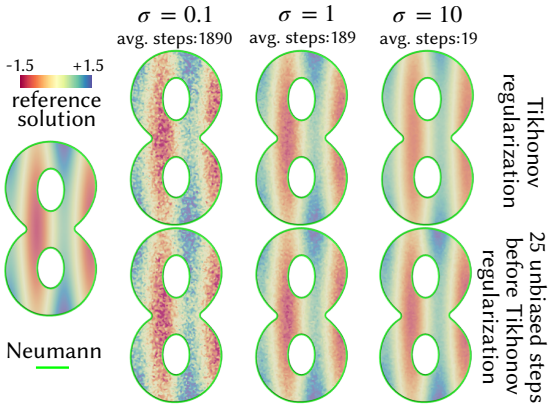


Figure 7.5: Top: A small Tikhonov parameter σ yields long walks and high variance, while larger σ values produce shorter walks with less noise but more bias. Bottom: Since the solution is often well-resolved by short walks, we apply regularization only to walks longer than a given length—yielding lesser noise and bias.

7.4 Adaptive Sampling And Denoising

Elliptic PDEs have highly regular solutions away from the source and boundary. Conventional methods exploit this behavior by interpolating over a mesh; we can likewise interpolate over scattered samples to dramatically reduce cost. For instance, in Fig. 7.7 (*center*), we use simple Poisson disk sampling [30] and *moving least squares (MLS)* interpolation [181]. To avoid "bleeding" artifacts, we shoot rays to exclude neighbors not visible from sample points. There are plenty of opportunities for improvement and acceleration (*e.g.*, via fast multipole methods [244]); note that *all* interpolation schemes introduce bias (including standard nodal interpolation in FEM).

We can also use adaptive sampling to concentrate effort on interesting regions. For instance, Fig. 7.7 (*right*) applies a simple scheme in the spirit of *irradiance gradients* [265]: we first estimate the solution and gradient at a set of seed points. Then, for any candidate sample x, we evaluate a 1st-order Taylor approximation $\widehat{u}(y_i) + \langle \widehat{\nabla u}(y_i), x - y_i \rangle$ for the k nearest neighbors $y_1, \ldots, y_k$ of x. If the sample variance (Eq. 3.4) divided by the mean is above a threshold $\alpha > 0$, we estimate the value and gradient at x and add it to the set. More sophisticated strategies from rendering provide ample inspiration for future work [288].

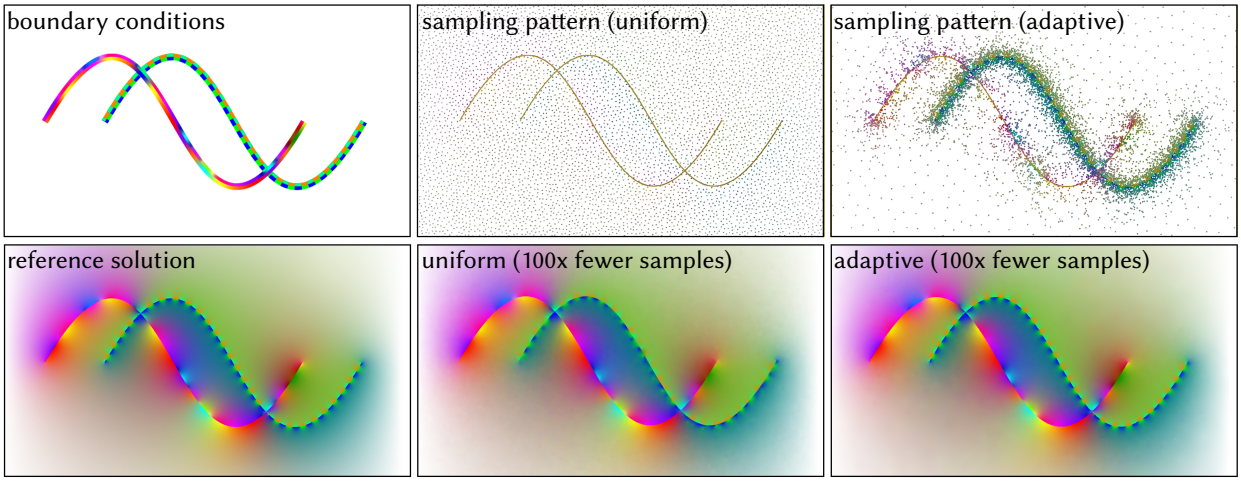


Figure 7.7: Just as mesh-based methods interpolate solution values at a few sparse points, we can rapidly visualize solutions to PDEs via scattered data interpolation. Here we solve a Laplace problem using either uniform or adaptive sampling and simple MLS interpolation. Adaptive sampling better resolves high-frequency boundary conditions.

Techniques for denoising rendered images [123, 137, 189, 190, 213, 226] also translate well to the PDE setting. For instance, in Fig. 7.6, we apply Intel's deep learning-based *Open Image Denoise* algorithm [112], using the boundary value at the closest point in place of the albedo map (no other results in this work use denoising). Training such a network on PDE data, rather than rendered images, should further reduce bias and improve performance.

Figure 7.6: Even for a small number of walks per pixel (left), techniques for denoising renders (center) closely match the reference solution (right), further increasing efficiency—especially for PDEs with smooth solutions.

7.5 Boundary Value Caching

Grid-free Monte Carlo methods such as walk on spheres independently estimate the solution at every point, and hence do not take advantage of the high spatial regularity of solutions to elliptic problems. Here we describe a fast caching strategy [166] for the boundary integral

$$\alpha(x)\, u(x) = \underbrace{\int_{\partial\Omega} P^{\mathbb{R}^d}(x,z)\, u(z) \;+\; G^{\mathbb{R}^d}(x,z)\, \frac{\partial u(z)}{\partial n_z}\, \mathrm{d}z}_{=:\, u_{\partial\Omega}(x)} + \underbrace{\int_{\Omega} G^{\mathbb{R}^d}(x,y)\, f(y)\, \mathrm{d}y}_{=:\, u_{\Omega}(x)} \tag{7.6}$$

of a (screened) Poisson equation in a domain Ω, where $P^{\mathbb{R}^d}$ and $G^{\mathbb{R}^d}$ are free space functions, rather than functions for the ball. To make use of this BIE, one must somehow determine the unknown boundary data: Dirichlet values u on the Neumann boundary $\partial\Omega_{\mathrm{N}}$, Neumann values $\partial u/\partial n$ on the Dirichlet boundary $\partial\Omega_{\mathrm{D}}$, and both, or one of, u and $\partial u/\partial n$ on the Robin boundary $\partial\Omega_{\mathrm{R}}$. Schemes such as the boundary element method use a finite-dimensional space of functions on the boundary (*e.g.*, basis functions associated with mesh nodes), and solve a dense, globally-coupled linear system for the best approximation to the true solution.

BVC takes a completely different approach, and instead uses WoSt to compute the unknown boundary values. It hence avoids global solves, boundary remeshing, and approximation of the function space; unlike BEM, it also handles the source term f. Moreover, as random walks can be expensive (especially in problems with predominantly Neumann boundaries), we cache these boundary values at a collection of random sample points along $\partial\Omega$. We then use a Monte Carlo estimate of Eq. 7.6 for cheap, output-sensitive evaluation of the solution (or its gradient) at any interior point $x \in \Omega$, without taking any further random walks (Sec. 7.5.1).

Overall, this scheme is easy to parallelize, and can be computed progressively (*e.g.*, for interactive preview). It can handle imperfect geometry (*e.g.*, with self-intersections) and detailed boundary/source terms without repairing or resampling the boundary representation (Fig. 8.15). We can also focus computation on a region of interest by caching points only on the boundary of a small subdomain $\Lambda \subset \Omega$ (Sec. 7.5.2)—unlike BEM which must perform a global solve over the entire boundary $\partial\Omega$.

In practice, we obtain far smoother results across the domain compared to directly using pointwise estimators like WoS or WoSt (Fig. 7.8, 7.9 & 7.10). This behavior can be attributed to correlations in the solution estimates at interior evaluation points that use the same boundary and source samples. On the flip side, error is now more global akin to traditional PDE solvers such as FEM and BEM (Fig. 7.16 & 7.17). Unlike pointwise estimators, we also observe boundary artifacts (Fig. 7.15), as samples are no longer generated in proportion to the singular functions $P^{\mathbb{R}^d}$ and $G^{\mathbb{R}^d}$. We show how to mitigate such artifacts in Sec. 7.5.3.

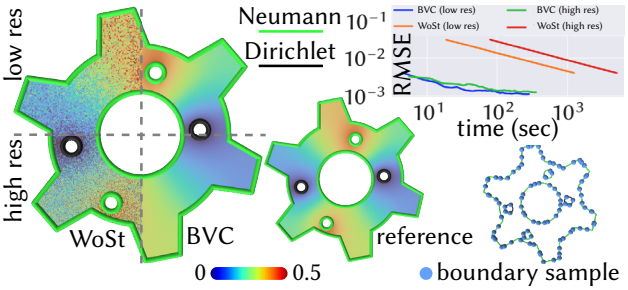


Figure 7.8: We obtain far smoother results with BVC compared to directly using pointwise estimators like WoSt at equal time. BVC uses the same boundary samples to determine the PDE solution across the domain.

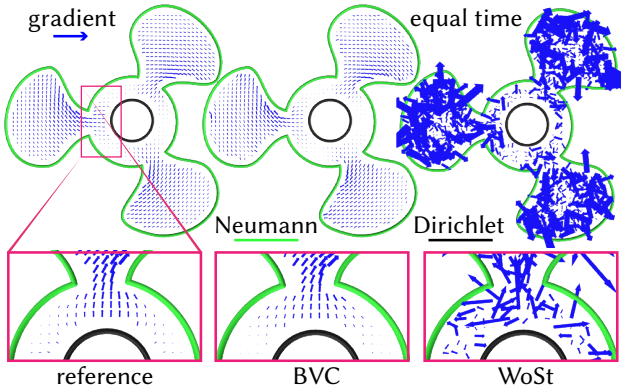


Figure 7.9: BVC gradients have considerably less noise compared to pointwise estimates, as they use known values of $\partial u/\partial n$ on the Neumann boundary. In contrast, WoSt gradients become noisier away from the Dirichlet boundary, as estimation requires longer random walks.

Sample Reuse In Rendering. Popular sample reuse schemes in rendering such as virtual point lights [46, 132], photon mapping [91, 92, 115] and ReSTIR [23, 193] share samples across pixels to amortize the cost of long ray traced paths, and inject global information into per-pixel radiance estimates. As a result, they are often more efficient at rendering scenes with complex geometry and lighting than brute-force path tracers.

BVC shares similarities with VPLs and photon mapping, in that samples generated and deposited on the scene (for us $\partial\Omega$) determine the radiance (*i.e.*, the solution u) over the image plane (which for us is either the entire domain Ω or a subset thereof). Unlike photon mapping however, BVC does not require an additional data structure like a *kd-tree* to store samples; instead we opt for a progressive formulation that discards boundary and source samples after splatting solution and gradient estimates in Ω. Similar to VPL methods, Monte Carlo noise is visually

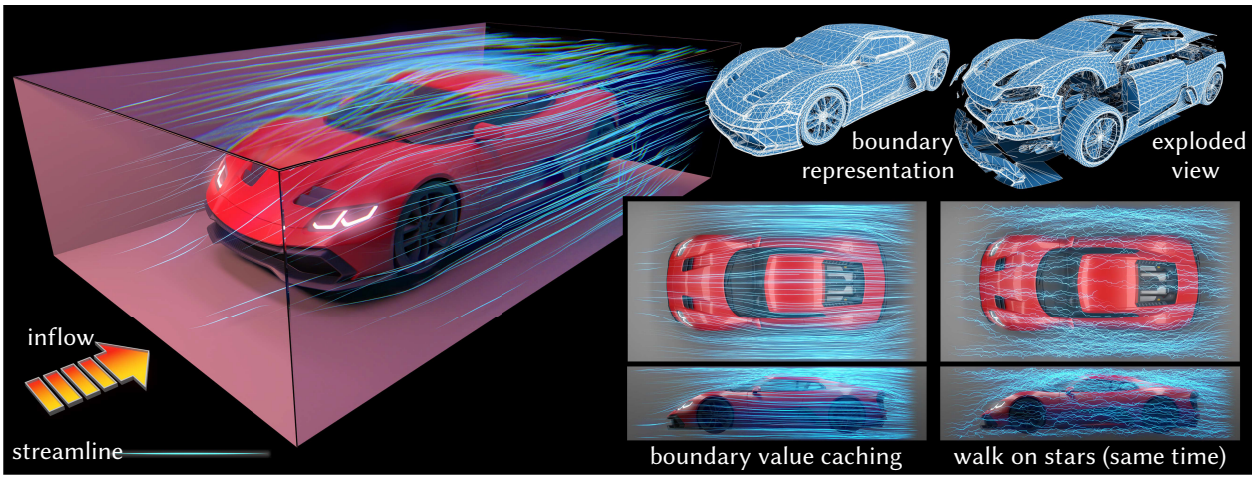


Figure 7.10: Boundary value caching dramatically reduces the total number of random walks needed to solve PDEs relative to pointwise Monte Carlo estimators. Here we show streamlines of a potential flow in a simulated wind tunnel, computed directly from a low-quality surface mesh originally intended for visualization rather than simulation.

suppressed [46, Fig. 1] from a combination of introducing correlations between estimates in Ω, and the smooth decay of the functions $P^{\mathbb{R}^d}$ and $G^{\mathbb{R}^d}$ away from the boundary (Fig. 2.3). VPLs are also prone to singularities, as sharing samples requires sacrificing perfect importance sampling [46, Sec. 5 & Fig. 9]; our artifact correction schemes take inspiration from similar techniques for VPLs [135]. Unlike VPL methods, we do not require testing for occlusion between deposited samples and evaluation points as the BIE contains no visibility term.

Techniques based on *lightcuts* [152, 264, 275] render scenes containing thousands of VPLs in real time. These methods cluster VPLs spatially in a tree, and then probabilistically select a subset of the VPLs that make the largest contribution at a given point. Akin to Fast Multipole and Barnes-Hut schemes [88, 202] for BEM, a lightcuts-based strategy should asymptotically reduce the quadratic complexity of evaluating the BIE with BVC. We leave development of such a strategy to future work.

7.5.1 Monte Carlo Estimation

We estimate the solution $u_{\partial\Omega} + u_\Omega$ in Eq. 7.6 at a set of evaluation points $\mathrm{evalPts} := \{x_k \in \Omega\}_{k=1}^K$ in a closed domain $\Omega \subset \mathbb{R}^d$ by creating two caches, $\mathrm{boundarySamples} := \{z_i, \widehat{u}(z_i), \widehat{\partial u/\partial n}(z_i)\}_{i=1}^N$ and $\mathrm{sourceSamples} := \{y_j, f(y_j)\}_{j=1}^M$, where N and M are user-specified cache sizes. The points z_i and y_j are sampled on the boundary $\partial\Omega$ and inside the domain Ω using probability densities $p^{\partial\Omega}$ and p^Ω respectively. The pointwise estimates $\widehat{u}(z_i)$ and $\widehat{\partial u/\partial n}(z_i)$ are computed using WoSt (Ch. 3, Sec. 3.2.5 & 7.2), while $f(y_j)$ are evaluations of the known source term.

Dirichlet-Neumann conditions. For BVPs with mixed Dirichlet and Neumann conditions, BVC then uses the two caches to form *correlated* Monte Carlo estimates of Eq. 7.6 at all points in $\mathrm{evalPts}$:

$$\widehat{u_{\partial\Omega}}(x_k) := \frac{1}{N}\sum_{i=1}^N \frac{P^{\mathbb{R}^d}(x_k, z_i)\ \widehat{u}(z_i)\ +\ G^{\mathbb{R}^d}(x_k, z_i)\ \widehat{\frac{\partial u}{\partial n}}(z_i)}{p^{\partial\Omega}(z_i)}, \tag{7.7}$$

$$\widehat{u_\Omega}(x_k) := \frac{1}{M}\sum_{j=1}^M \frac{G^{\mathbb{R}^d}(x_k, y_j)\ f(y_j)}{p^\Omega(y_j)}. \tag{7.8}$$

ALGORITHM 7: A *boundary value caching* strategy to evaluate a d-dimensional boundary integral equation inside a closed user-defined boundary $\partial\Lambda$

struct BoundarySample
  $z, n_z \leftarrow$ null ▷*Sample location and unit outward normal on* $\partial\Lambda$
  $\widehat{u}, \widehat{\frac{\partial u}{\partial n}} \leftarrow 0$ ▷*Estimates of solution and normal derivative*
struct EvaluationPoint
  $x \leftarrow$ null ▷*Location for evaluating the BIE*
  $\widehat{u}^{\text{sum}}_{\partial\Lambda}, \widehat{u}^{\text{sum}}_{\Lambda} \leftarrow 0$ ▷*Running sums for solution evaluation*
  $N, M \leftarrow 0$ ▷*Boundary and source sample count*
  function GetSolution()
    return $\widehat{u}^{\text{sum}}_{\partial\Lambda}/N \;+\; \widehat{u}^{\text{sum}}_{\Lambda}/M$

Input: A set of evalPts, boundary and source cache sizes N and M, nWalks for pointwise estimation, and PDFs $p^{\partial\Lambda}$ and p^{Λ} for sample generation.
Output: An updated solution estimate at each evaluation point.
function UpdateSolution(evalPts, N, M, nWalks, $p^{\partial\Lambda}$, p^{Λ})
  boundarySamples $\leftarrow$ BoundarySample$[N]$ ▷*Initialize* N *boundary samples*
  parallel for b **in** boundarySamples **do**
    $b.z, b.n_z \sim p^{\partial\Lambda}$ ▷*Generate boundary sample from PDF* $p^{\partial\Lambda}$
    $b.\widehat{u}, b.\widehat{\frac{\partial u}{\partial n}} \leftarrow$ WalkOnStars$(b.z, b.n_z, \text{nWalks})$ ▷*Use* nWalks *to estimate* u *and* $\partial u/\partial n$ *with WoSt*
  for b **in** boundarySamples **do**
    $z, n_z \leftarrow b.z, b.n_z$
    parallel for e **in** evalPts **do** ▷*Splat boundary contribution from* b *to all* evalPts
      $e.\widehat{u}^{\text{sum}}_{\partial\Lambda} \mathrel{+}= \left(P^{\mathbb{R}^d}(e.x, z, n_z) b.\widehat{u} + G^{\mathbb{R}^d}(e.x, z) b.\widehat{\frac{\partial u}{\partial n}}\right) / p^{\partial\Lambda}(z)$
      $e.N \mathrel{+}= 1$
  for j **in** Range(M) **do**
    $y \sim p^{\Lambda}$ ▷*Generate source sample from PDF* p^{Λ}
    parallel for e **in** evalPts **do** ▷*Splat source contribution from* y *to all* evalPts
      $e.\widehat{u}^{\text{sum}}_{\Lambda} \mathrel{+}= G^{\mathbb{R}^d}(e.x, y) f(y) / p^{\Lambda}(y)$
      $e.M \mathrel{+}= 1$

In App. A, we provide expressions for $P^{\mathbb{R}^d}$ and $G^{\mathbb{R}^d}$ for the Poisson and screened Poisson equations. Alg. 7 provides pseudocode, and App. B.2 discusses the extension to open domains and double-sided boundary conditions.

Robin conditions. For BVPs with Robin conditions, we need not estimate both u and $\partial u/\partial n$ with WoSt. Instead, substituting $\partial u/\partial n = \ell - \mu u$ on a Robin boundary $\partial\Omega_{\text{R}}$ in Eq. 7.6 gives

$$\begin{aligned} \alpha(x)\; u(x) = &\int_{\partial\Omega_{\text{R}}} \left(P^{\mathbb{R}^d}(x,z) - \mu(z) G^{\mathbb{R}^d}(x,z)\right) u(z)\,\mathrm{d}z \\ &+ \int_{\partial\Omega_{\text{R}}} G^{\mathbb{R}^d}(x,z)\; \ell(z)\,\mathrm{d}z \;+\; \int_{\Omega} G^{\mathbb{R}^d}(x,y)\; f(y)\,\mathrm{d}y, \end{aligned} \tag{7.9}$$

for which only the solution u needs to be estimated on $\partial\Omega_{\text{R}}$. As an alternative, we could also make the substitution $u = (\ell - \partial u/\partial n)\,/\mu$ when $\mu > 0$, and instead solve for unknown values of $\partial u/\partial n$ on $\partial\Omega_{\text{R}}$. We observe that estimating $\partial u/\partial n$ typically works better when μ is large. This is because BVC does not importance sample the term $P - \mu G$ in Eq. 7.9 when generating its cache samples on $\partial\Omega_{\text{R}}$, which means that a large μ amplifies noise in estimated values of u on $\partial\Omega_{\text{R}}$.

Gradient Estimation. We can reuse *the same* cached boundary and source samples to also form Monte Carlo estimates of the solution gradient at each evaluation point (Fig. 7.9):

$$\widehat{\frac{\partial u_{\partial\Omega}}{\partial x}}(x_k) := \frac{1}{N}\sum_{i=1}^{N} \frac{\frac{\partial P^{\mathbb{R}^d}}{\partial x}(x_k, z_i)\ \widehat{u}(z_i)\ +\ \frac{\partial G^{\mathbb{R}^d}}{\partial x}(x_k, z_i)\ \widehat{\frac{\partial u}{\partial n}}(z_i)}{p^{\partial\Omega}(z_i)}, \tag{7.10}$$

$$\widehat{\frac{\partial u_{\Omega}}{\partial x}}(x_k) := \frac{1}{M}\sum_{j=1}^{M} \frac{\frac{\partial G^{\mathbb{R}^d}}{\partial x}(x_k, y_j)\ f(y_j)}{p^{\Omega}(y_j)}. \tag{7.11}$$

We provide expressions for the derivatives of $P^{\mathbb{R}^d}$ and $G^{\mathbb{R}^d}$ in App. A.

Sampling. We can use a discrete *cumulative density function (CDF)* table [203, Sec. 13.3] with stratified random numbers to generate boundary samples over elements (*e.g.*, triangles) of a polygonal mesh. Faster sample generation is possible with an alias table [262, 263]. Source samples can likewise be generated uniformly inside Ω with stratified sampling [203, Sec. 13.8]. We use this sampling setup for all BVC figures in this text except Fig. 7.10, where we find that results improve significantly if the boundary samples in Eq. 7.7 & 7.10 are weighted by the area of their associated Voronoi cell. Similar area weighting strategies have proven effective for surface reconstruction [12, Fig. 5], and yield a consistent Monte Carlo estimator that provides provably better convergence [89].

Progressive Evaluation. BVC is progressive in two ways. First, we can improve estimation quality at a set of evaluation points, by generating new caches and using them to update existing estimates (Alg. 7, UpdateSolution). Second, we can compute solution estimates at new evaluation points by iterating over existing caches.

Bias. Assuming the pointwise estimates $\widehat{u}(z_i)$ and $\widehat{\partial u/\partial n}(z_i)$ are unbiased, the estimators in Eq. 7.7–7.11 are also unbiased via the linearity of expectation. In reality, most pointwise estimators have a small amount of controllable bias from the ε-shell (Sec. 6.3). Unlike MLS interpolation and adaptive sampling (Sec. 7.4), BVC does not introduce any additional bias, while improving efficiency noticeably (Fig. 7.8).

7.5.2 Boundary Specification

When the solution needs to be evaluated within a localized region Λ inside the domain Ω (Fig. 7.11), we specialize the BIE to this region by generating source samples in Λ and boundary samples on $\partial\Lambda$ (Alg. 7, *lines* 13 & 21). We use uniform densities $p^{\Lambda} = 1/|\Lambda|$ and $p^{\partial\Lambda} = 1/|\partial\Lambda|$ for sample generation, though we could use densities specialized to a specific PDE to reduce variance further. The solution integrates to 0 outside Λ by construction.

When the solution needs to be evaluated within the entire domain Ω (*i.e.*, $\Lambda \equiv \Omega$), we incorporate the known boundary data $\partial u/\partial n = h$ on $\partial\Omega_N$ directly into our sample estimates,

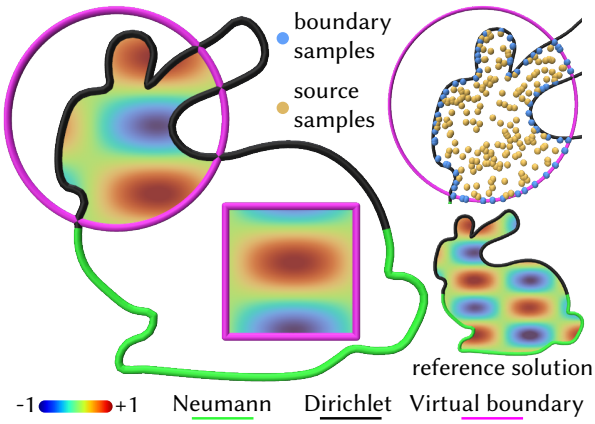


Figure 7.11: We can focus computation on a region of interest by caching points on the region boundary.

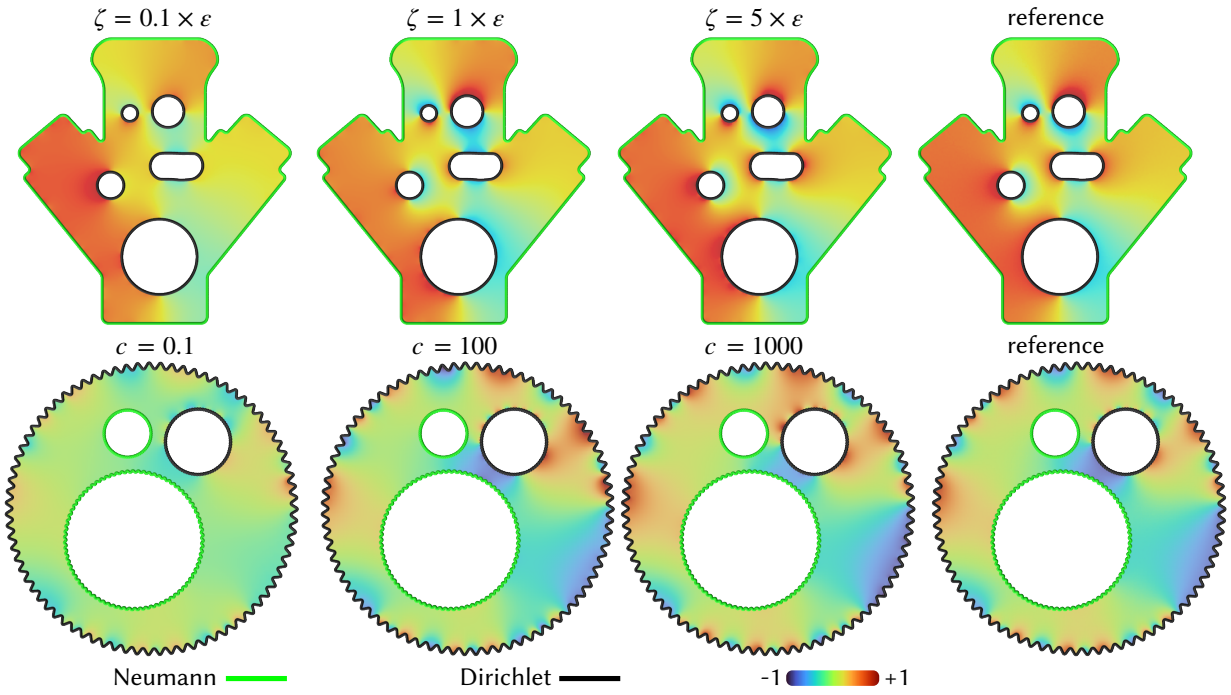


Figure 7.12: Top: Setting the ζ offset parameter to $0.1 \times \varepsilon$ effectively sets each Dirichlet boundary sample's $\partial u/\partial n$ estimate to zero, as balls centered at each sample point are contained entirely inside the epislon shell—this biases the solution estimate inside the domain. Bias diminishes with increasing offset values. Bottom: Smaller values of the clamp c for the Poisson kernel (Eq. 7.12) suppress singular artifacts near the boundary but bias interior estimates without our correction strategy, shown here on a model scaled to fit inside a unit sphere.

rather than estimating it from scratch. Unfortunately, estimating $\partial u/\partial n$ on the Dirichlet boundary $\partial\Omega_D$ is challenging, as WoSt requires a ball with a non-zero radius to estimate the solution gradient (Eq. 3.18). Instead of dealing with $\partial\Omega_D$ directly, we define a closed region bounded by the Neumann boundary $\partial\Omega_N$ and an offset Dirichlet boundary $\partial\Omega_D^\zeta$ where $\zeta > \varepsilon$ (Fig. 7.13). We then generate boundary samples on $\partial\Omega_D^\zeta$, and estimate both u and $\partial u/\partial n$. Moreover, we separately use WoSt to compute the solution at any evaluation point that is within a distance ζ to $\partial\Omega_D$, as random walks typically terminate quickly when started close to the Dirichlet boundary.

There are two considerations involved in choosing an offset ζ: the minimal feature size of the domain Ω, and the amount of bias that can be tolerated in the estimate for $\partial u/\partial n$ on $\partial\Omega_D^\zeta$, based on its proximity to the epsilon shell $\partial\Omega_D^\varepsilon$. We do not define sample reuse regions in the vicinity of thin features—instead, we create multiple disconnected regions inside Ω where boundary and source samples are cached, while using WoSt to estimate the solution pointwise elsewhere. We use $\zeta = 5 \times \varepsilon$ in our implementation to balance between bias and variance in our estimates for $\partial u/\partial n$ on $\partial\Omega_D^\zeta$ (Fig. 7.12, *top* shows an ablation). We leave a more principled unbiased estimation of this quantity directly on $\partial\Omega_D$ to future work.

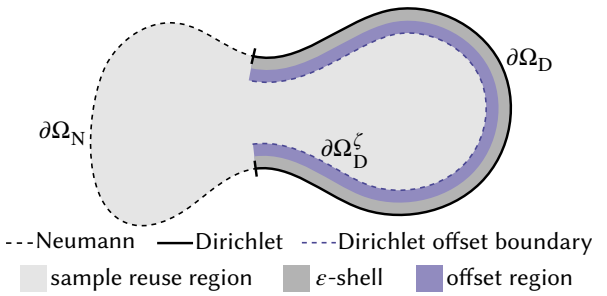


Figure 7.13: To evaluate the BIE inside Ω, BVC generates samples on the Neumann boundary $\partial\Omega_N$ and an offset Dirichlet boundary $\partial\Omega_D^\zeta$ where $\zeta > \varepsilon$.

7.5.3 Singularities

Though the free space Green's function and its derivatives decay smoothly, they are singular at the point they are centered on (Fig. 2.3). Therefore, BVC can suffer from local artifacts, especially near the boundary, as it uses uniformly distributed boundary and source samples to evaluate Eq. 7.7–7.11. In contrast, such artifacts do not arise with WoSt as the corresponding functions for a ball are importance sampled, *i.e.*, $p^{\partial B} \propto P^{\mathrm{B}}$ and $p^{B} \propto G^{\mathrm{B}}$.

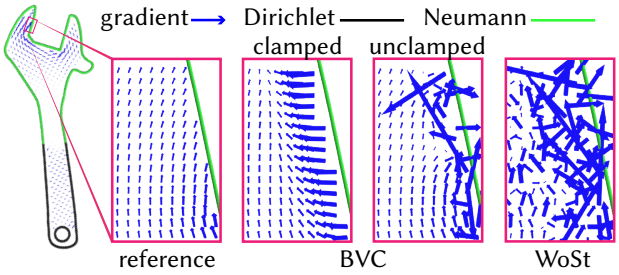


Figure 7.14: Naïvely clamping singular kernels near the boundary suppresses noise, but introduces bias.

Similar artifacts are often suppressed in virtual point light methods by clamping the singular geometry term in the rendering equation [46, Sec. 5 & Fig. 9]. However, clamping can introduce noticeable bias in the solution estimate with both VPLs and BVC (Fig. 7.12). Likewise, clamping derivative norms in Eq. 7.10–7.11 leads to smoothly-varying but biased gradients near the boundary (Fig. 7.14). Below, we extend Kollig and Keller [135]'s bias correction strategy for VPLs to the functions $P^{\mathbb{R}^d}$ and $G^{\mathbb{R}^d}$, and leave the extension to gradient estimators to future work.

Removing Bias From Clamping $P^{\mathbb{R}^d}$. The function $P^{\mathbb{R}^d}$ has a large value when the points x and z are close to each other. To mitigate artifacts that arise from not importance sampling points z from $P^{\mathbb{R}^d}$, we rewrite the first term in Eq. 7.6 over a user-defined region Λ as follows:

$$\int_{\partial\Lambda} P_c^{\mathbb{R}^d}(x,z)\, u(z)\, \mathrm{d}z \;+\; \int_{\partial\Lambda} \left[P^{\mathbb{R}^d}(x,z) - P_c^{\mathbb{R}^d}(x,z)\right] u(z)\, \mathrm{d}z, \tag{7.12}$$

where c is a positive user-specified clamping parameter, and $P_c^{\mathbb{R}^d} := \max(-c,\ \min(c, P^{\mathbb{R}^d}))$. As before, we use uniformly distributed boundary samples on $\partial\Lambda$ to estimate the first integral in Eq. 7.12 at any evaluation point $x \in \Lambda$. We choose the bound c based on the scale of the scene (Fig. 7.12, *bottom* shows an ablation), though strategies for automatically setting this parameter can likely be adapted from the VPL literature [135, Sec. 2.3].

To estimate the second integral, we can use direction sampling to importance sample new boundary samples on $\partial\Lambda$, since the free space Poisson kernel $P^{\mathbb{R}^d}$ defines a signed solid angle over the boundary just like the Poisson kernel P^{B} for a ball (see Sec. 4.2.2 and App. A.1 for details). We then use WoSt to estimate unknown solution values u at intersection points z (of which there might be more than one if $\partial\Lambda$ is nonconvex). However, we only need to estimate u when $P_c^{\mathbb{R}^d}(x,z) \neq P^{\mathbb{R}^d}(x,z)$, *i.e.*, when the evaluation point x is in close proximity to an intersection point $z \in \partial\Lambda$. This is typically not the case for most evaluation points inside Λ, as $P^{\mathbb{R}^d}(x,z)$ falls off quickly (Fig. 2.3). As shown in the inset, we can estimate Eq. 7.12 in both an artifact- and bias-free manner by running only a few random walks for the second integral.

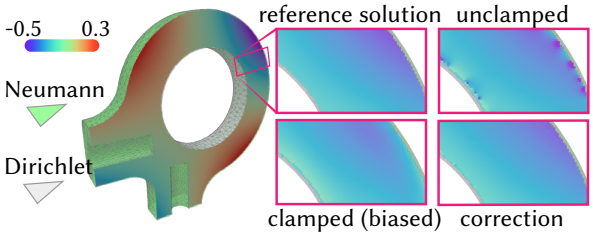


Figure 7.15: Our clamping correction strategy (bottom right) effectively suppresses local artifacts from singular kernels near the boundary without bias (top right).

Removing Bias From Clamping $G^{\mathbb{R}^d}$. We follow a similar recipe to mitigate localized artifacts arising from not importance sampling the Green's function in the BIE. In particular, we rewrite the source integral over a region Λ as follows:

$$\int_\Lambda G_c^{\mathrm{B}}(x,y)\, f(y)\, dy \;+\; \int_\Lambda \left[G^{\mathrm{B}}(x,y) - G_c^{\mathrm{B}}(x,y)\right] f(y)\, dy, \tag{7.13}$$

where rather than the free space Green's function $G^{\mathbb{R}^d}$, we now use the Green's function of a ball G^{B}, and $G_c^{\mathrm{B}} := \min(-c,\, \max(c, G^{\mathrm{B}}))$. We use G^{B} in place of $G^{\mathbb{R}^d}$ since it can be importance sampled for points y (Sec. 7.1). Moreover, both the clamping and sampling described in the previous section remain unchanged, as $-\partial G^{\mathrm{B}}/\partial n =: P^{\mathrm{B}} \equiv P^{\mathbb{R}^d}$. We choose the ball $\mathrm{B}(x, R)$ to be centered at the evaluation point x such that it contains the region Λ inside it. The first integral in Eq. 7.13 is then estimated as usual with uniformly distributed source samples in Λ. The second integral is estimated by drawing y from G^{B}, and setting $f = 0$ if $y \notin \Lambda$.

7.5.4 Convergence

In Fig. 7.8 & 7.17, we show that even in 2D, BVC provides up to an order of magnitude error reduction at equal time compared to WoSt for mixed boundary value problems. The reason is twofold. First, runtime efficiency improves as we do not perform independent random walks for interior evaluation points. Second, boundary samples inject global information about the solution into interior estimates. The overall result is that BVC reduces the dimensionality of a PDE solve. In contrast, BEM trades not discretizing the domain with solving a much denser linear system.

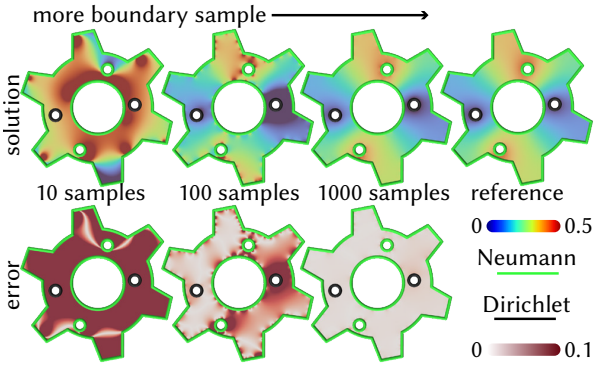


Figure 7.16: Akin to traditional PDE solvers, BVC demonstrates a global error in the solution profile that vanishes with more boundary samples.

Similar to traditional PDE solvers requiring global solves, error in the interior now depends on the number of boundary and source samples used. Fig. 7.16 & 7.17 show that error vanishes with more samples, though we note that Eq. 7.7–7.11 provide unbiased estimates even with just a single sample. We also observe lower interior error with more accurate estimates for u and $\partial u/\partial n$ on the boundary. As derivative estimates are typically noisier than solution estimates (Fig. 7.4), we perform more random walks for $\widehat{\partial u/\partial n}$ on an offset Dirichlet boundary $\partial\Omega_{\mathrm{D}}^{\zeta}$ (Sec. 7.5.2), compared to $\widehat{u}$ on $\partial\Omega_{\mathrm{N}}$. As the default, we take $10\times$ more walks for boundary samples on $\partial\Omega_{\mathrm{D}}^{\zeta}$ than on $\partial\Omega_{\mathrm{N}}$. In general, the overhead of taking more walks is small, as walks starting close to the Dirichlet boundary are usually much shorter.

7.6 Reverse Random Walks

In place of estimating u or $\partial u/\partial n$ on the boundary as with BVC, Qi et al. [206] derive a bidirectional formulation for WoS that can simulate random walks in "reverse". These reverse walks splat known Dirichlet and source data into the interior of the domain Ω. Like bidirectional algorithms for light transport [140, 253], reverse walks can be more efficient than typical "forward" walks to the boundary (Sec. 3.2), as a single reverse walk contributes to the solution estimate at multiple

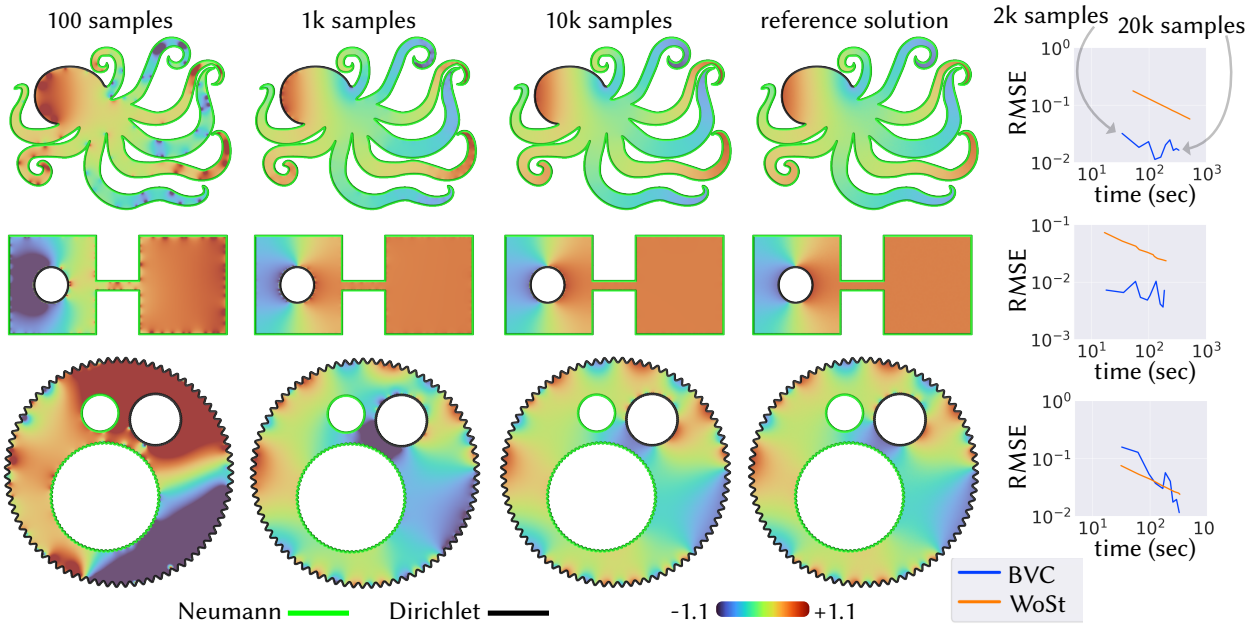


Figure 7.17: BVC amortizes the cost of long walks in Neumann dominated problems (top two rows) and hence improves efficiency over point estimators like WoSt. However, in more Dirichlet dominated problems with higher frequency boundary conditions (bottom row), efficiency drops due to shorter walk lengths, not importance sampling the singular kernels in the BIE, and noise in the estimates of $\partial u/\partial n$ on the boundary.

points in Ω. Compared to BVC, we observe empirically that Qi et al. [206]'s approach can also have lower variance at equal time for pure Dirichlet problems (see inset), as it avoids the need to estimate $\partial u/\partial n$ on the boundary and evaluate $P^{\mathbb{R}^d}$ during BIE estimation. We refer to Qi et al. [206] for further details.

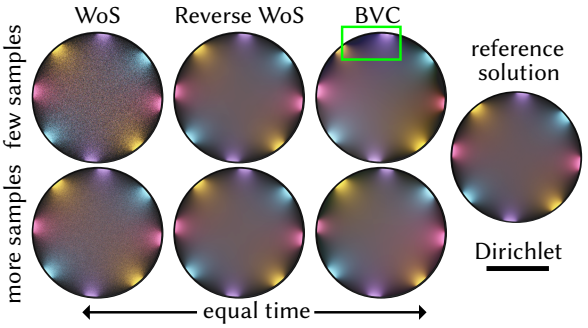


Figure 7.18: BVC can have higher variance than Qi et al. [206]'s specialized approach for Dirichlet conditions.

Here we generalize reverse walks to Robin (and Neumann) problems [167]. We first substitute $\partial u/\partial n = \ell - \mu u$ into Eq. 2.10 as in Sec. 4.3, but use the sets $A = \Omega$ and $C = \Omega$ instead. This yields

$$\alpha(x)\, u(x) = \int_{\partial\Omega_R} G^\Omega(x,z)\, \ell(z)\, \mathrm{d}z \;+\; \int_\Omega G^\Omega(x,y)\, f(y)\, \mathrm{d}y, \tag{7.14}$$

where the Green's function of the domain G^Ω is the *fundamental solution* to the PDE

$$\begin{aligned} \Delta G^\Omega(x,y) &= \delta_x(y) && \text{on } \Omega, \\ \tfrac{\partial G^\Omega(x,y)}{\partial n_y} + \mu(y) G^\Omega(x,y) &= 0 && \text{on } \partial\Omega_R. \end{aligned} \tag{7.15}$$

Compared to Eq. 4.6 or Eq. 7.9, there are no unknown values u or $\partial u/\partial n$ in the boundary integral in Eq. 7.14, as the Robin boundary condition on G^Ω cancels them out. However, to compute the solution u anywhere in Ω, we must now solve for G^Ω at points $z \in \partial\Omega_R$ and $y \in \Omega$. We describe how to estimate G^Ω with WoSt next—and use the resulting reverse walks starting from z or y to improve solution estimates at points $x \in \Omega$. As with BVC, we observe smoother results, albeit

with slightly more local correlation artifacts at low sample counts (see Fig. 7.19), and less control over where the solution is evaluated inside Ω.

Estimating G^Ω with WoSt. We require an integral expression for G^Ω on a star-shaped region St to use WoSt. Following Sec. 4.3, we have for any two points $x \in \Omega$ and $z \in \partial\Omega_R$,

$$\begin{aligned} G^\Omega(x,z) &= \int_{\partial \mathrm{St}(x,R)} \rho_\mu(x,z')\; P^B(x,z')\; G^\Omega(z',z)\,\mathrm{d}z' \;+\; \int_{\mathrm{St}(x,R)} G^B(x,y)\;\delta_z(y)\,\mathrm{d}y \\ &= \int_{\partial \mathrm{St}(x,R)} \rho_\mu(x,z')\; P^B(x,z')\; G^\Omega(z',z)\,\mathrm{d}z' \;+\; G^B(x,z) \\ &= \int_{\partial \mathrm{St}(z,R)} \rho_\mu(z,z')\; P^B(z,z')\; G^\Omega(z',x)\,\mathrm{d}z' \;+\; G^B(z,x), \end{aligned} \tag{7.16}$$

where the third equality follows from G^Ω being symmetric. This expression can be used to define random walks that start from randomly sampled points on $\partial\Omega_R$ with known Robin data ℓ; for the source term f, walks will instead start from Ω. However, unlike the "forward" estimator from Eq. 4.8 which gathers information about boundary conditions and sources during a walk, a "reverse" WoSt estimator for G^Ω will instead allow for a single walk to splat ℓ and f values at more than one point inside Ω (similar to Qi et al. [206]'s reverse WoS walks for pure Dirichlet problems). In more detail, for $k \geq 0$, we estimate G^Ω using the following recursive single-sample estimator for Eq. 7.16:

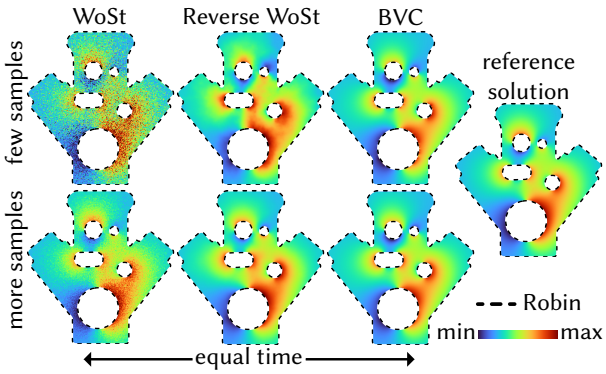


Figure 7.19: Like BVC, reverse walks provide smoother results with lesser noise on Robin problems than WoSt.

$$\widehat{G}^\Omega(x,z_k) = \rho_\mu(z_k,z_{k+1})\;\widehat{G}^\Omega(z_{k+1},x) \;+\; G^B(z_k,x). \tag{7.17}$$

Here $G^B(z_k,x) > 0$ when $x \in \mathrm{St}(z_k,R_k)$, and zero otherwise. As in Sec. 4.2.2, we use direction sampling to determine the next walk location, *i.e.*, $z_{k+1} \in \partial\mathrm{St}(z_k,R_k) \sim P^B(z_k,z_{k+1})$. We also use Russian roulette to terminate walks with probability $\rho_\mu(z_k,z_{k+1})$, as described in Sec. 4.3.4.

To then solve Poisson equations with Robin conditions, we estimate Eq. 7.14 using

$$\widehat{u}(x) = \frac{\widehat{G}^\Omega(x,z_0)\;\ell(z_0)}{\alpha(x)\;p^{\partial\Omega_R}(z_0)} + \frac{\widehat{G}^\Omega(x,y_0)\;f(y_0)}{\alpha(x)\;p^\Omega(y_0)}, \tag{7.18}$$

where we are free to sample points $z_0 \in \partial\Omega_R$ and $y_0 \in \Omega_R$ from densities $p^{\partial\Omega_R}$ and p^Ω (respectively) of our choosing. Finally, we use Eq. 7.18 and the random walks defined by Eq. 7.17 to evaluate $\widehat{u}$ at points x, as long as these points are contained inside star-shaped regions $\mathrm{St}(z_k,R_k)$ centered at walk locations z_k. This approach also applies to Neumann conditions when $\mu = 0$.

7.7 Weight Window

For variable coefficient problems, we can design specialized variance reduction strategies based on the specific form of the Monte Carlo estimator in Eq. 5.10, where the value of the estimate

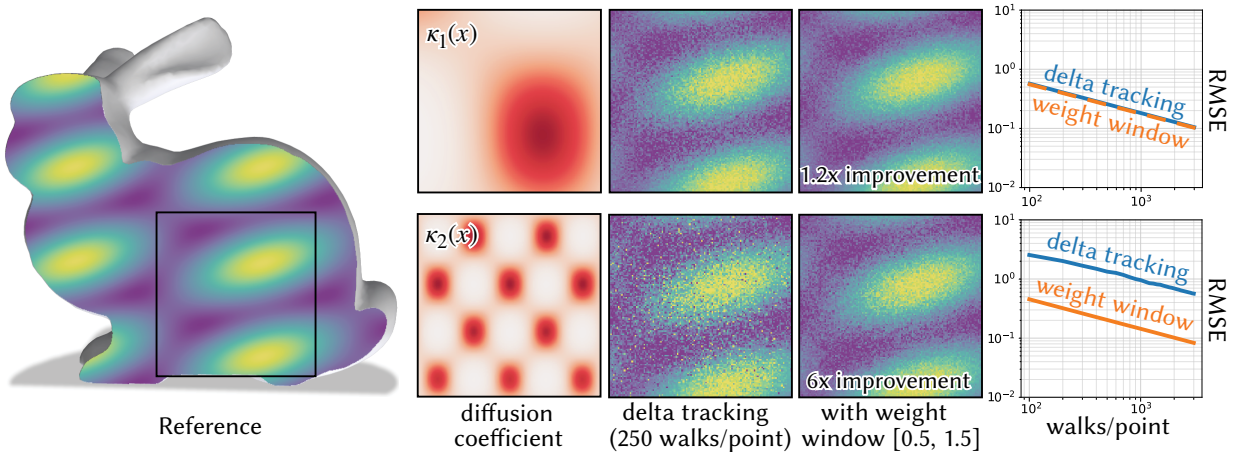
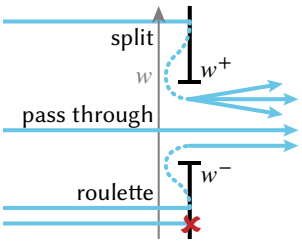


Figure 7.20: Weight windows significantly reduce variance for problems with large variation in coefficients, as seen here for the problem $\nabla \cdot (\kappa(x)\nabla u) = -f(x)$ with two different diffusion functions $\kappa_1(x), \kappa_2(x)$.

$\widehat{u}(x_k)$ is multiplied by a weight in $[0, \infty)$ at every step x_k of a walk. For instance, in the delta tracking variant of WoS, $\widehat{u}$ is scaled either by $(1 - \sigma'(y_{k+1})/\overline{\sigma}) \cdot \sqrt{\kappa(y_{k+1})/\kappa(x_k)}$ or $\sqrt{\kappa(z_{k+1})/\kappa(x_k)}$, based on whether we sample the volume or the boundary term respectively (see Alg. 4). These weights serve as a source of variance in the estimator (Fig. 7.20). This is especially true for large values of the bounding parameter $\overline{\sigma}$, since a high average number of steps per walk can result in a walk throughput that is either very small or very large. We use a commonly employed variance reduction strategy from neutron transport [102] called a *weight window* to address this issue, as it keeps throughput roughly constant for any walk.



A weight window achieves variance reduction though a combination of Russian roulette and splitting (Sec. 3.1.1). The former terminates walks with small weights, as it is often not worthwhile completing walks whose likely contribution to the estimator is small. The latter splits walks with large weights into multiple new equally weighted walks. This prevents any single walk from accumulating a large throughput, while encouraging better exploration of the domain through newly spawned walks. We employ a simple version of this strategy that uses a static window size $[w^-, w^+]$, where w^- and w^+ are the minimum and maximum throughput values allowed for any walk (we use $[0.5, 1.5]$ in our implementation). A walk whose throughput w at any step lies within the window is allowed to continue without modification. The walk is then terminated using a Russian roulette survival probability of w/w^- when $w < w^-$. Otherwise, it is split into $m := w/w^+$ new walks, each with a weight of w/m, when $w > w^+$. Since m is generally not an integer, we apply the *expected value splits* approach of Booth [25] which uses $n := \lfloor m \rfloor$ walks with probability $n + 1 - m$, and $n + 1$ walks otherwise.

Fig. 7.20 highlights the effectiveness of a static weight window in reducing variance in problems with high frequency coefficients and a large bounding parameter $\overline{\sigma}$. In the neutron transport literature, specifying the window size adaptively has been shown to provide improvements in computational efficiency over a static window by up to an order of magnitude [26, 256, 258]. We leave this optimization to future work.

Chapter 8

Evaluation & Comparisons

As noted in Ch. 1, a key motivation for developing grid-free Monte Carlo methods is to push simulation methods closer to the geometric complexity seen in photorealistic rendering—and in nature. In Sec. 8.1, we mock up example problems from scientific, geometric, and visual computing to stress-test our solvers. Importantly, these examples do not aim to model exact physics or make quantitative predictions—we seek only to examine solver performance in the presence of (i) complex geometry, (ii) different boundary conditions, and (iii) varying material coefficients, all of which are ubiquitous in real physical problems.

In Sec. 8.2, we then contrast the Monte Carlo approach with more conventional grid-based techniques for linear elliptic PDEs. The literature on solving these problems is vast, so we only discuss broad classes of solvers such as finite element methods and finite differences, rather than specific algorithms. Given the complementary strengths and tradeoffs, a likely outcome in the long run is that there are problems and algorithms from the FEM/FD setting that do not naturally translate to Monte Carlo methods, and likewise, paths that can easily be taken via Monte Carlo but not via traditional PDE solvers. We conclude this chapter with Sec. 8.3, wherein we demonstrate that solvers like WoS and WoSt can have orders of magnitude less error in their solution estimates than other grid-free Monte Carlo strategies for linear elliptic PDEs.

8.1 Geometric Robustness, Scalability & Flexibility

We use a multicore CPU-based implementation for the majority of the experiments described below, and achieve essentially linear scaling—developing an optimized GPU implementation is an important avenue of future work. Unless otherwise noted, all experiments used a 64-core 3rd Generation Intel Xeon workstation with 64 GB RAM.

8.1.1 Heat Transfer And Thermal Analysis

Heat transfer is a central topic in thermal engineering, with three basic modes: radiation, conduction, and convection. Thermal radiation is well-captured by 1st-order Monte Carlo light transport simulation, whereas conduction and convection involve diffusion, which must be simulated via a 2nd-order method like WoS or WoSt [14]. Thermal convection can also include turbulent advection, *i.e.*, Navier-Stokes, which is not considered here but can be solved in part via WoS [211].

Accurate thermal analysis of complex geometry is central to the success of a wide variety of engineering problems, ranging from the design of printed circuit boards [32] and residential architecture [126] to spacecrafts and robotics. For instance, NASA advocates for use of detailed thermal analysis throughout the design process—instructing its engineers that *"thermal modeling is required beginning at the project conceptual design stage and continuing through preliminary and detailed design stages ... simplified calculations and rules of thumb are useful at this stage, but a computer model ... provides the ability to evaluate and respond quickly to proposed system trade-offs."* [180]. Just as computer graphics has long enjoyed the ability to iterate on illumination for virtual environments

(via Monte Carlo rendering), the solvers we develop in this work can help engineers achieve the same kind of fast and quantitatively reliable feedback during the design process—rather than waiting on bottlenecks like mesh generation (Fig. 1.1).

Mars Rover. Fig. 1.9 mocks up a representative use case of our method in a geometrically complex scenario: thermal analysis of NASA's *Curiosity* Mars rover. In particular, we compute the steady-state temperature on the rover surface by solving a Laplace equation with Robin boundary conditions. Robin boundary data is given by thermal radiation from the sun, computed via ordinary ray tracing. Since we do not have access to original NASA schematics for *Curiosity*, we use an artist-generated model, using texture values to set Robin boundary conditions. From the perspective of simulation, however, there is nothing special about this model—it could trivially be swapped out with the true engineering model (or any other candidate design). The use of partially-absorbing Robin conditions provides the opportunity for far more accurate physical modeling than purely absorbing (Dirichlet) or reflecting (Neumann) conditions alone. More accurate simulation might be obtained by coupling our WoSt solver with one that models, *e.g.*, convective heat transfer [14], though the low density of the Martian atmosphere makes this term largely negligible [255].

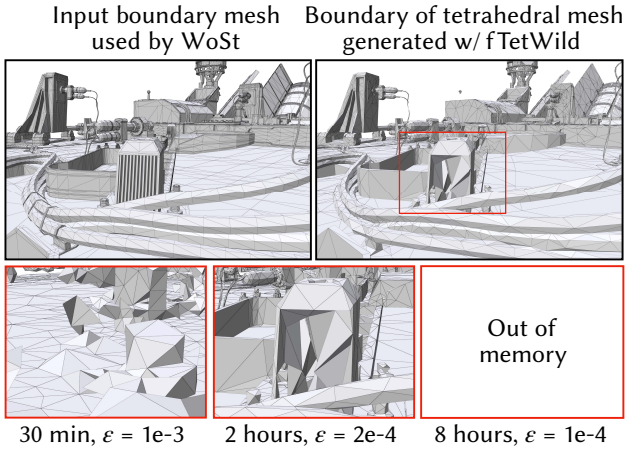


Figure 8.1: Generating tetrahedral meshes for accurate FEM simulation can be challenging, as meshing tools either fail to capture important detail in the input model, or routinely run out of memory at finer tolerances. Here, we run fTetWild [105] on the Mars Rover from Fig. 1.9. TetGen [234] is unable to tetrahedralize this model as it contains self-intersections.

Fig. 1.9 (*top right*) illustrates a *deferred shading* approach [50] which is quite natural in the Monte Carlo setting, but has not been considered in prior work on walk on spheres methods. Rather than evaluating the solution at every point of a regular grid, or every vertex of the boundary mesh, we first render the Cartesian xyz coordinates of the model as seen from a viewpoint of interest (Fig. 1.9, *top center of top right*). Each pixel in this coordinate image is then used as the starting point for random walks via WoSt. In this way, we only spend time computing the solution at points that actually need to be inspected for analysis—Fig. 1.9 (*bottom*) shows a collection of closeup viewpoints solved in the same fashion. Moreover, since Monte Carlo accumulates a running sum, we can immediately visualize a rough estimate of the solution that progressively improves over time (Fig. 1.9, *bottom row of top right*). Across all four viewpoints, we compute the solution at 793k points in each image; for this model, each walk took about 0.48 milliseconds.

Attempting to capture the domain geometry with a finite element mesh leads to extreme compute times and, ultimately, failure, even with state-of-the-art robust meshing software (Fig. 8.1). Overall, the use of deferred shading, plus the fact that we avoid volumetric meshing, makes this approach orders of magnitude faster than any finite element approach—offering a qualitative shift in the approach to engineering design.

Toast. Inspired by toast-darkening experiments of Myhrvold and Migoya [178], Fig. 4.2 models heat transfer from a toaster to a piece of bread, represented by a CT scan with 3.9 million

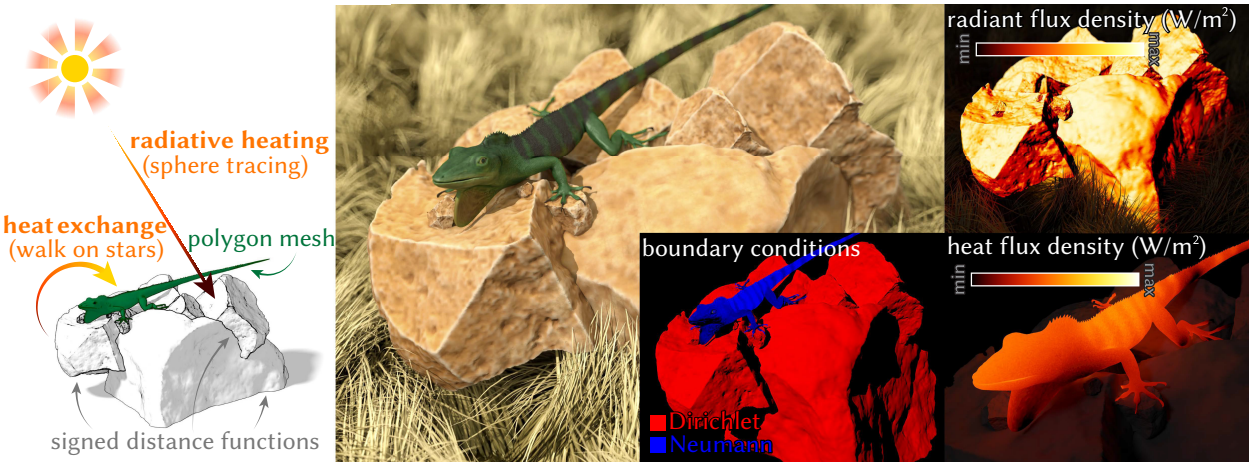


Figure 8.3: Like Monte Carlo ray tracing, WoSt enables flexible modeling of scenarios that are geometrically and physically complex. Center: Here an ectotherm (anolis carolinensis) warms itself on a rock, which in turn is heated by the sun. Left: WoSt can mix and match different geometric representations—here a polygon mesh and a signed distance function. In this case, it is also easily combined with ray tracing used to determine Dirichlet boundary conditions for the heat transfer problem. Right: Unlike grid-based PDE techniques, WoSt can handle highly-detailed boundary conditions without needing to resolve them on a mesh—here the lizard's texture controls the rate of heat absorption via Neumann boundary conditions, yielding stronger warming along the dark stripes.

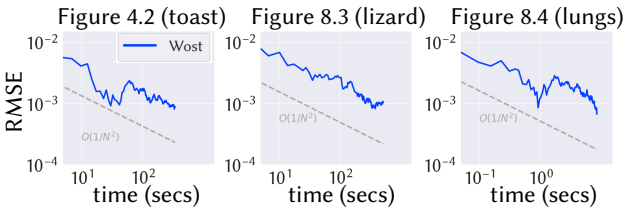


Figure 8.2: WoSt exhibits the expected rate of convergence for a Monte Carlo estimator, shown here for 8 fixed points on examples from Sec. 8.1. Reference solutions are also computed via WoSt with 2^{16} walks per point, as there is no analytical solution and no feasible alternatives to compute it. Timings were taken on an 8 core M1 MacBook Pro.

boundary elements. To model diffusive conduction, we solve a Laplace equation with large and small Dirichlet values on the heating elements and toaster cavity (respectively), and Neumann conditions on the bread. The solution is evaluated at roughly 2 million boundary points, using 1 walk per point for fast preview and 256 walks for the final solution; on average, WoSt takes 0.166 milliseconds per point for each walk (Fig. 8.2 plots error versus time). A simple phenomenological model is used to translate surface temperature into color (though more principled models of *Maillard browning* could be used here [36]). We observe a marked difference between the temperature distribution resulting from radiation and conduction—emphasizing the necessity of 2nd-order models for accurate thermal predictions.

Ectothermic Lizard. Fig. 8.3 shows another heat transfer experiment, where Dirichlet conditions induced by solar radiation are used to determine heat absorbed by an ectothermic lizard, modeled via detailed spatially-varying Neumann inflow conditions. Unlike FEM or BEM (Sec. 8.2) where boundary data must be evaluated ahead of time, Dirichlet data is evaluated *on demand* via sphere tracing [97]. Scene geometry is represented by a 1.2 million element boundary mesh (for the lizard) and implicit signed distance functions (for the rocks), highlighting the ability of WoSt to work with mixed boundary representations without global meshing. The solution is evaluated at 285k boundary points using 1024 walks per point, taking on average 0.121 milliseconds per point for each walk. As in Monte Carlo rendering (and unlike FEM/BEM), scene setup required no model conversion or cleanup—even though data was pulled directly from the internet.

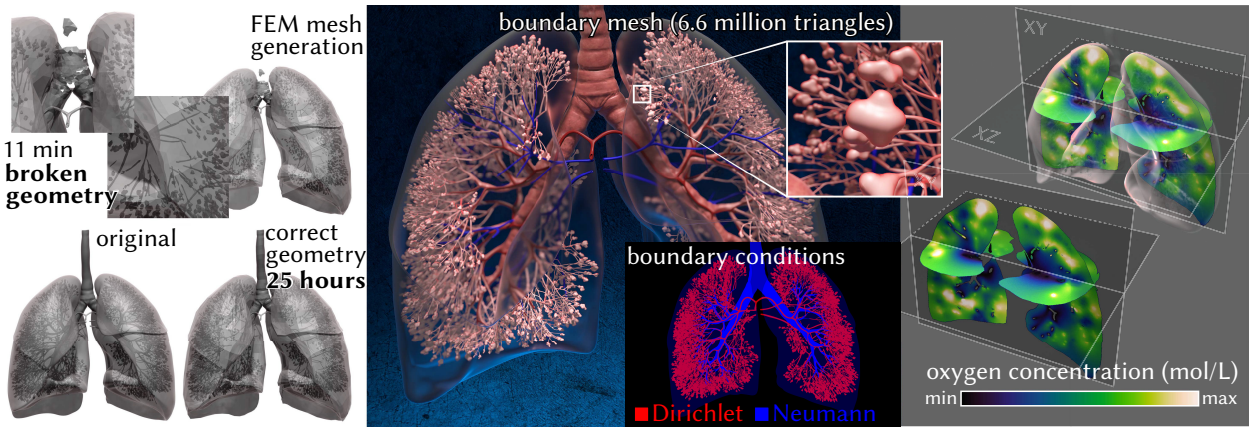


Figure 8.4: Where will oxygen flow at the beginning of a breath? Here we use WoSt to simulate gas exchange via Laplacian transport [86], directly on a detailed lung model with thin features (center). The output-sensitivity of our method enables us to focus computation purely on the slice planes used for visualization (right), rather than needing to solve over the whole domain. Attempting simulation on the same model using traditional solvers leads to significant problems, either because meshing destroys critical details (top left), or takes more than 25 hours to produce a mesh that captures the original geometry (bottom left). In contrast, WoSt provides near-immediate feedback that reliably reflects the true geometry and solution.

Blackbody Emitters. Monte Carlo methods are popular in rendering because they provide immediate feedback that can be progressively improved, enabling artists and engineers to iterate on designs quickly. We explored this modality for variable-coefficient PDEs by implementing a GPU version of our delta tracking WoS estimator (Sec. 5.2.1) in the *Unity* game engine, where domain geometry is encoded by a signed distance function (SDF), and visualized via sphere tracing. This setup allows us to interactively explore problems of immense geometric complexity—for instance, Fig. 5.1 shows a scene with an infinite, aperiodic arrangement of detailed models (achieved via instancing [53, 238]) and high-frequency material coefficients, which would be impractical or impossible for conventional PDE solvers. In fact, as shown in Fig. 8.10, generating a volume mesh for even a small region of this scene is prohibitively expensive. Fig. 6.2 shows interactive editing via CSG operations, which are performed easily on SDFs. The solution is visualized only on parts of a 2D slice visible from the camera, and samples are accumulated progressively until the scene changes—in this case our solver runs at 60 frames per second.

8.1.2 Laplacian Transport

Fig. 8.4 models oxygen diffusion in the lungs, one of many *Laplacian transport* phenomena with mixed boundary conditions [86] (other examples include electric transport in electrolytic cells [67, 218], diffusion of reactive molecules towards catalytic surfaces in heterogeneous catalysis [219], and diffusion of spins in confining porous media in nuclear magnetic resonance [87]). We make a simplification by using Neumann rather than Robin boundary conditions, though the latter are easily supported with WoSt. While conventional methods can solve such problems, meshing is both a major performance bottleneck and a hindrance for end-to-end robustness. In this case, even a state-of-the-art method [105] yields badly broken geometry; tweaking parameters to capture the correct geometry incurs a full day of compute time, eliminating any advantage of a fast solve. With WoSt we get feedback reliably and immediately in an output-sensitive fashion, here restricted to a cross section. In particular, we evaluate the solution on a 512×512 grid using 1024 walks per point; WoSt takes on average 0.021 milliseconds per point for each walk.

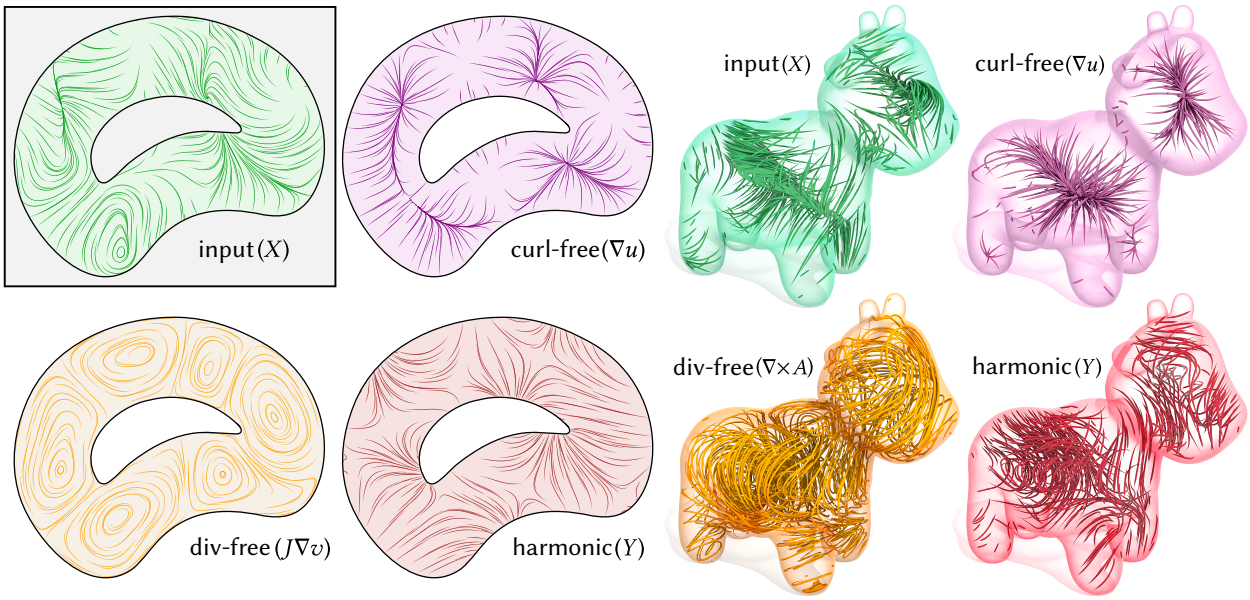


Figure 8.5: Helmholtz decomposition on 2D (left) and 3D (right) domains. Here we decompose an input vector field X given by a closed-form, analytic expression into its curl-free, divergence-free, and harmonic components. With the Monte Carlo approach, each streamline can be traced independently in parallel, without having to discretize X on a background mesh, or solve a linear system.

8.1.3 Vector Field Processing

A vector-valued solution X can be visualized using very sparse sampling. For instance, to draw a standard quiver plot, we need only compute one solution value per arrow—rather than solving the PDE over the entire domain. We can also draw streamlines γ, obtained by numerically integrating the ordinary differential equation $\frac{d}{dt}\gamma(t) = X(\gamma(t))$ starting at some collection of seed points. Once again, the value of X need only be estimated at the sparse sample points used by the ODE integrator—examples computed via Huen's method are shown in Fig. 7.10 & 8.5.

Helmholtz Decomposition. A powerful tool in flow processing and visualization is the *Helmholtz decomposition*, which expresses a given vector field X as the sum of a curl-free, divergence-free, and harmonic part. In computer graphics, such decompositions have been considered on meshes [247, 279] and point clouds [210], both of which entail discretizing space and solving large linear systems. On a domain $\Omega \subset \mathbb{R}^3$, one possible decomposition is

$$X = \nabla u + \nabla \times A + Y, \tag{8.1}$$

where u is the solution to the scalar Poisson equation $\Delta u = \nabla \cdot X$, A is the solution to the vector Poisson equation $\Delta A = \nabla \times X$ (which is just three componentwise scalar equations), and Y is the remaining part. Setting u and A to zero along $\partial\Omega_D$ yields standard *normal-parallel* boundary conditions [21, Sec. 4.2], which ensure the decomposition is unique [20]. In 2D we get an analogous decomposition by replacing A with a scalar potential $a : \Omega \to \mathbb{R}$, and replacing $\nabla \times A$ with $J\nabla a$, *i.e.*, a 90-degree rotation of the gradient of a.

To compute this decomposition via Monte Carlo, we compute the necessary derivatives (Sec. 3.2.5) of the solution to the two Poisson equations. Basic 2D and 3D examples are shown Fig. 8.5, where we trace streamlines by applying a standard ODE integrator to the derivative estimates.

Potential Flow. Given the spatial smoothness of solutions to elliptic problems, a key strength of the boundary value caching method from Sec. 7.5 is its ability to suppress the salt-and-pepper noise characteristic of independent Monte Carlo estimates. In Fig. 7.10, we observe noticeably smoother results from correlated sample estimates of the gradient, where we solve for streamlines of a potential flow with Dirichlet boundary conditions of -1 and 1 at the front and back of the tunnel respectively, and Neumann conditions of 0 elsewhere. Similar to the Helmholtz decomposition, we use Huen's method to draw streamlines along the estimated gradient direction. Steamlines start at a collection of random seed points in the domain, and can be regenerated cheaply once estimates for u and $\partial u/\partial n$ have been computed on the boundary. Notably, we do not require a high-quality simulation mesh for this task.

8.1.4 Diffusion Curves and Surfaces

To examine the ability of our solver to handle complex boundaries and boundary conditions, we implemented *diffusion curves* (Fig. 6.1) and *diffusion surfaces* (Fig. 8.6), which encode a resolution-independent 2D image or volumetric texture as the solution to a Poisson equation with two-sided Dirichlet boundary conditions. A nice feature of Monte Carlo is that we can directly compute the solution on a zoom-in or cross section to evaluate points of interest, rather than precomputing a global solution (as in Orzan et al. [192]). Moreover, the Monte Carlo approach completely decouples signal frequency from geometric resolution—*e.g.*, Fig. 7.7 shows highly oscillatory boundary conditions on just two cubic Bézier segments. In contrast, traditional methods must explicitly refine a background grid or mesh to even *encode* such boundary conditions (much less compute the solution).

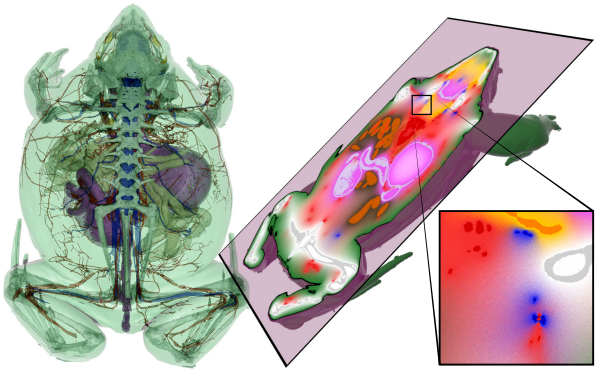


Figure 8.6: Left: a 3.3M triangle mesh with fine features, from a CT scan of hemisus guineensis (courtesy Blackburn Lab). To visualize the solution to a volumetric PDE, we can significantly reduce cost by computing only a 2D slice of the full 3D solution (right). Inset: fine features are perfectly preserved as we work with the exact input geometry.

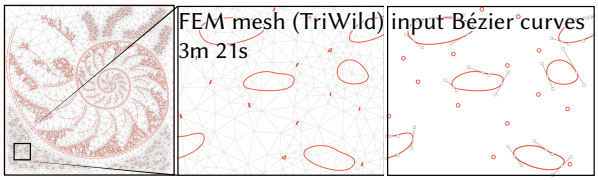


Figure 8.7: Even in 2D, robust meshing algorithms can distort small geometric detail.

To further examine tradeoffs with grid-based approaches, we generated triangular and tetrahedral meshes (respectively) for these two domains using TriWild [104] and fTetWild [105], which are state-of-the-art robust meshing algorithms. In addition to needing significant time and memory just for mesh generation, the default tolerances in these methods were not sufficient to capture fine-scale details. Tuning such tolerances is expensive since a mesh must be regenerated each time, which is hard to do in an automatic fashion. Moreover, tolerances that are too tight can lead to an explosion in cost—for instance, even just lowering fTetWild's default tolerance of 10^{-3} to 10^{-4} used up 22GB of memory and failed to generate a volume mesh for Fig. 8.6 after 14 hours of wall clock time.

In contrast, the ε tolerance for Dirichlet problems with WoS has very little effect on cost and accuracy (Sec. 6.3): for very loose tolerances small features can get rounded out, but cannot disappear (Fig. 6.4). Even tiny tolerances add only a few steps to each walk, due to the $\log(1/\varepsilon)$ behavior of the WoS algorithm, *i.e.*, computation cost does not blow up dramatically.

Walk on Curved Surfaces. A basic hypothesis of the original WoS algorithm is that a random walk exits every point on the boundary of a ball with equal probability (Sec. 2.2.1). However, on surfaces with non-constant curvature this hypothesis no longer holds: intuitively, more walks will escape through the "valleys" than through the "mountains". As a result, standard WoS cannot be used for many algorithms in geometric and scientific computing that need to solve equations on a surface.

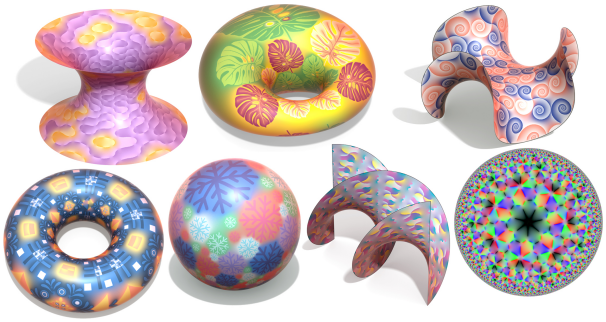
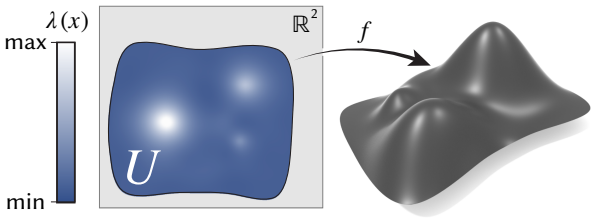


Figure 8.8: Variable coefficients enable us to extend WoS to curved domains (not previously possible). Here we directly resolve intricate boundary conditions for diffusion curves—without generating a fine surface mesh that conforms to boundary curves or a spatially-adaptive grid in the parameter domain.



Our variable-coefficient scheme enables WoS to be applied to curved surfaces for the first time. In particular, consider any surface expressed as a *conformal parameterization* $f : \mathbb{R}^2 \supset U \to \mathbb{R}^3$; conformal means that f distorts the surface by a positive scaling $\lambda(x)$ at each point $x \in U$, *i.e.*, $J_f^T J_f = \lambda(x) I$, where I is the identity and J_f is the Jacobian of f. The Laplacian Δ_f of the curved surface is then related to the ordinary Laplacian via $\Delta = \lambda \Delta_f$. Hence, we can solve PDEs on the curved surface by replacing the usual diffusion coefficient $\kappa(x)$ with $\lambda(x)$. Fig. 8.8 shows several examples; for periodic domains (like the torus) our walks simply "wrap around." In theory, *every* surface admits a conformal parameterization (by the uniformization theorem [2]), but in practice many important surfaces used in engineering (such as NURBS or other spline patches) are expressed in non-conformal coordinates. To directly handle such patches, we would need to extend our method to *anisotropic* diffusion coefficients—an important topic for future work.

8.2 Comparison With Traditional Grid-Based PDE Solvers

There are many methods for solving linear elliptic PDEs—far too many to review and compare with here. Instead, we examine the unique benefits of Monte Carlo in terms of the basic properties shared by all grid-based methods within each major class. Since our goal is to directly resolve fine details in the boundary conditions and coefficients, we omit discussion of *homogenization* schemes, which approximate fine-scale behavior via coarse-scale models [1, 56, 57, 60, 157]; Monte Carlo methods like ours may in fact help to accurately determine parameters for such schemes. We also do not consider recent learning-based approaches such as *Physics-informed neural networks* [208], but note that one could use our solvers to generate synthetic data for training, or design hybrid strategies for supersampling and denoising [151].

8.2.1 Finite Element Methods

All finite element methods (including BEM and meshless FEM which we discuss below) adopt a common framework: for a linear PDE $Lu = f$, find the best approximation to u in a finite-dimensional function space. *E.g.*, standard *Galerkin FEM* uses an approximation $\widehat{u} := \sum_{i=1}^{n} u_i \phi_i$

in a basis $\phi_1, \ldots, \phi_n : \Omega \to \mathbb{R}$, where $u_i \in \mathbb{R}$ are unknown coefficients. Letting $\langle u, v \rangle := \int_\Omega u(x)v(x)\,\mathrm{d}x$ denote the L^2 inner product, one then seeks a $\widehat{u}$ satisfying

$$\langle L\widehat{u}, \phi_j \rangle = \langle \widehat{f}, \phi_j \rangle, \quad j = 1, \ldots, n, \tag{8.2}$$

i.e., such that $\widehat{u}$ agrees with the true solution u when restricted to the subspace $V := \mathrm{span}(\{\phi_i\})$. To solve Eq. 8.2, one rewrites it as

$$\textstyle\sum_{i=1}^n u_i \langle L\phi_i, \phi_j \rangle = \sum_{i=1}^n f_i \langle \phi_i, \phi_j \rangle. \tag{8.3}$$

The inner products on the left- and right-hand side define *mass* and *stiffness* matrices, and are often further approximated via numerical quadrature. From this perspective, the only difference between flavors of FEM is the choice of basis functions ϕ_i (and the difficulty of integrating them). Otherwise, all finite element methods share a common set of challenges:

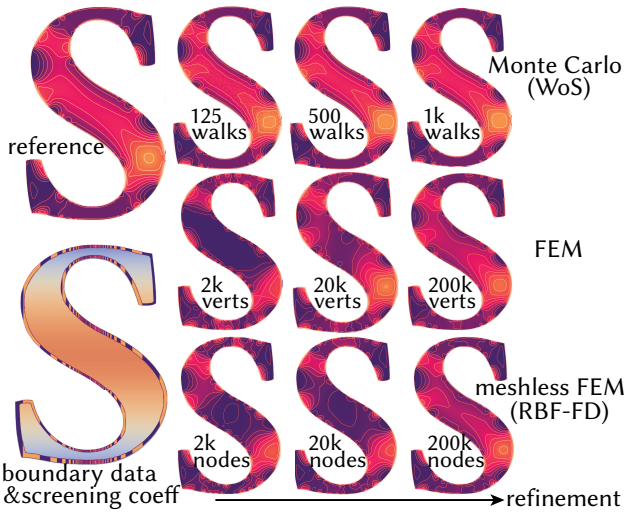


Figure 8.9: With both FEM and meshless FEM, local aliasing of high-frequency boundary data yields large global errors in the solution, demanding significant refinement. In contrast, WoS methods always capture the global solution (even with very few samples); error instead manifests as high-frequency noise.

- They must all solve a globally coupled system of equations.
- They are all prone to spatial aliasing in the geometry, solution, boundary conditions, source terms, and/or coefficients, since any finite basis $\{\phi_i\}$ provides limited spatial resolution.
- They all demand spatial discretization (meshing or sampling) to define bases ϕ_i, which can be costly and error prone.

In contrast, Monte Carlo methods like WoS can directly evaluate the solution at any point without meshing or global node placement, and without a global solve. Moreover, they do not suffer from aliasing in the solution or problem data, since functions are not restricted to a finite-dimensional subspace V (see Fig. 8.9). Though sharing information between nodes (via a linear system) with FEM improves efficiency, it is difficult to make an exact performance comparison between WoS algorithms and traditional solvers in terms of target accuracy, since performance is contingent on several factors beyond just rates of convergence. Crucially, the basic $O(1/\sqrt{N})$ convergence rate of Monte Carlo provides a measure of error relative to *other methods that estimate integrals* (such as Gauss quadrature)—and not those that solve PDEs, where factors like the cost of mesh generation and refinement, parallel communication overhead and the need for local versus global evaluation of the solution are far more significant in practice.

Mesh-Based FEM

The basic premise of FEM is to reduce a PDE to a finite dimensional system of equations. Constructing this system necessitates sampling or tessellating the domain or its boundary, making discretization error inevitable. Most often, FEM bases ϕ_i are defined via polyhedral mesh elements. Quickly and robustly meshing large, detailed and/or imperfect geometry (*e.g.*, with self-intersections) is an ongoing "grand challenge," where even the most advanced methods [103, 104, 105] can struggle (Fig. 1.2, 1.3, 8.1 & 8.4). This problem gets harder if the mesh must also

be refined for spatially varying coefficients: even with intelligent *adaptive mesh refinement (AMR)* [285, 286], meshing quickly becomes prohibitive (Fig. 8.10). More recent *a priori* p-refinement does not help, since it considers only element quality, and not spatial frequencies in the solution or problem data [227].

On the other hand, a key feature of grid-free Monte Carlo methods is that solution accuracy has nothing to do with the quality of mesh elements, which greatly improves robustness. WoS algorithms bypass meshing entirely and need only a BVH-based spatial hierarchy for geometric queries (Ch. 6), which uses little memory (Fig. 8.11) and can be built in a fraction of a second (Fig. 1.8), even for totally degenerate geometry (Fig. 1.7, *right*); sharp edges, small details, and thin features in the geometry are preserved *exactly*. These savings are compounded for problems with dynamic geometry, where one can just update or re-fit the BVH [136]. As with rendering, we can achieve extreme geometric complexity at low memory cost via techniques like instancing [238] or procedural modeling [53].

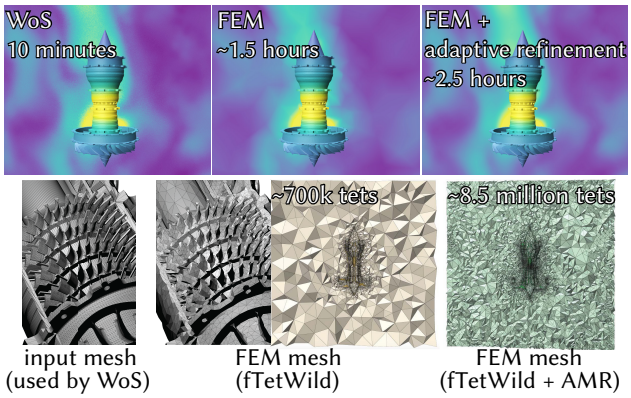


Figure 8.10: Even when starting with a coarse approximation of just a single blackbody from Fig. 5.1, FEM takes immense time and memory to resolve detailed variations due to PDE coefficients. Here, initial coarse meshing by fTetWild [105] takes about 1.5 hours to produce a mesh that cannot resolve fine details in the solution or geometry. After 1 hour more of AMR via Anderson et al. [4], the solution is better resolved, but the mesh size has blown up. Our CPU-based WoS implementation takes about 10 minutes total, on the same machine.

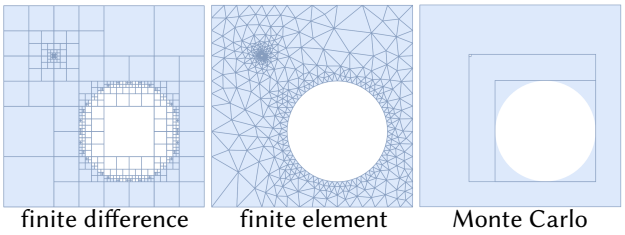


Figure 8.11: Grid- and mesh-based algorithms (left, center) consume significant memory since they explicitly discretize even simple geometry, and must use gradual grading to maintain good element quality. Monte Carlo methods (right) concisely represent smooth geometry and small features via a BVH, providing excellent memory scaling.

Moreover, proving consistency with FEM is challenging even for fairly common equations [59, 241]. Schemes that are consistent in a weak sense can still give unreliable *pointwise* derivatives [221, Fig. 26], which are needed for tasks like curvature evaluation [165]. Using higher-order elements can also significantly inflate degree—on triangle meshes, for instance, 5th-order polynomials are needed to get even C^1 continuity [195], and standard subdivision bases are not C^2 at irregular vertices [201, Sec. 1.8]. In addition, FEM convergence can slow down in the presence of features like *reentrant corners* [82, Eq. 2], which are common in real-world geometry. Since WoS algorithms exactly simulate stochastic processes that model smooth PDEs, such issues simply do not arise, but one must contend with noise instead.

Meshless FEM

Though *meshless FEM (MFEM)* seems like a natural alternative to grid-free Monte Carlo methods, the term "meshless" is a bit of misnomer: MFEM does not need a polyhedral mesh, but must still discretize the domain by carefully arranging a collection of nodes (Fig. 1.5). Nodes are then associated with bases ϕ_i, such as *radial basis functions (RBFs)*, to build mass and stiffness matrices. To couple bases with overlapping support, one must identify neighbors, and form a global graph structure similar in size and complexity to a polyhedral mesh. Node locations must satisfy

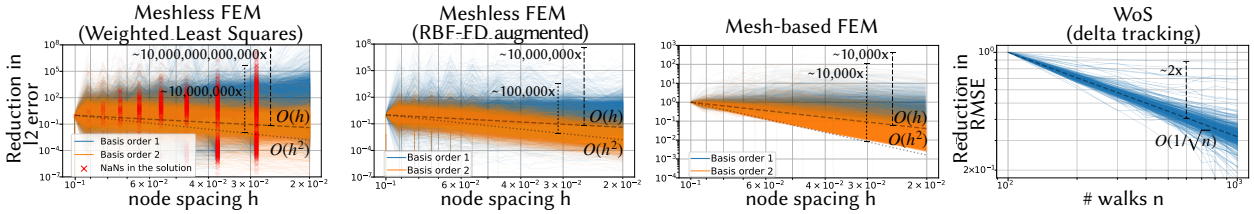


Figure 8.14: Convergence of meshless FEM is not well-understood: such methods often fail to converge under refinement and/or show extremely large variation in error, as seen here for a variable-coefficient problem on models from the Thingi10k dataset [280]. Both mesh-based FEM and Monte Carlo methods come with rigorous convergence guarantees and behave much more predictably under refinement.

criteria that can be just as difficult and delicate to enforce as mesh quality criteria [149, Ch. 3], and often require global optimization akin to mesh smoothing [237]. Moreover, just as one bad element can ruin an FEM solution, bad node placement can lead to catastrophic failure (*e.g.*, NaNs in the solution; see Fig. 8.14, *left*). One can adaptively sample nodes to mitigate spatial aliasing, but unlike mesh-based FEM, adaptive refinement for MFEM is poorly understood and lacks rigorous convergence guarantees. Typical MFEM bases are also *approximating* rather than *interpolating*, complicating enforcement of boundary conditions [73, 185]; some methods hence modulate bases by distance-like functions [229], but still effectively discretize functions on the interior by choosing a finite basis $\{\phi_i\}$.

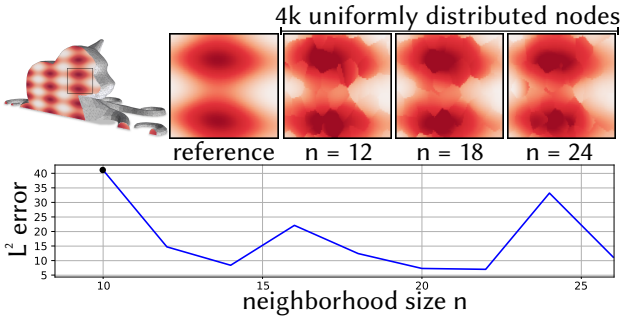


Figure 8.12: In practice, it can be difficult to find reliable parameters for MFEM, e.g., increasing neighborhood size can increase error in unpredictable ways. In contrast, WoS algorithms require little-to-no parameter tuning.

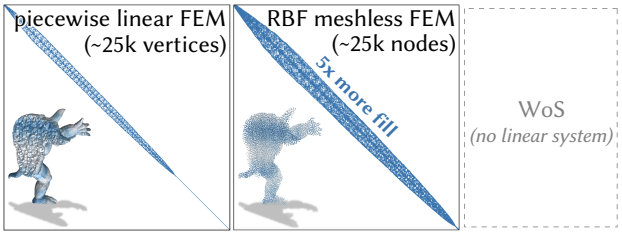


Figure 8.13: MFEM must solve a much denser linear system than even standard FEM, whereas WoS avoids solving a global system altogether.

As shown in Fig. 8.14, a more serious challenge with MFEM is *stagnation*: only until recently [15, 16, 69], MFEM might fail to converge without careful problem-specific tuning of parameters like neighborhood size. Unfortunately, increasing neighborhood size does not always make the solution better (see Fig. 8.12). Moreover, whereas convergence of *adaptive* FEM is rigorously understood [162], there is a dearth of corresponding results for adaptive MFEM schemes, which is especially important for problems with detailed geometry and coefficients. In practice, MFEM also requires denser matrices (Fig. 8.13) than those used in mesh-based FEM, while often providing less accurate results. For instance, methods such as *RBF-FD with polynomial augmentation* [69], which converge under refinement, require at least order-2 bases.

On the whole, MFEM is not known for its reliability. In stark contrast, the WoS algorithms we develop here guarantee that the expected solution equals the true solution of an elliptic PDE with minimal parameter tuning. Moreover, unlike MFEM, these Monte Carlo methods are truly "meshless": at no point does one require a global sampling or meshing of the domain. Finally, though MFEM has been around for a long time, it has not seen nearly as much use in practice as mesh-based methods (*e.g.*, with very few open-source or commercial packages available).

Boundary Element Methods

BEM approximates the solution using basis functions ϕ_i associated only with elements of a boundary mesh (such as free-space Green's functions). BEM draws a natural comparison with grid-free PDE solvers like WoS and BVC, since it is specifically designed to solve BIEs (Sec. 2.2), and need not discretize the interior of the domain. However, there is a significant difference in capabilities: whereas WoS easily handles problems with source terms and spatially varying coefficients on the domain interior, basic BEM ignores these terms altogether (see Fig. 1.4). In order to handle general interior terms, one must couple BEM with a second interior solver such as FEM, meshless FEM, or finite differences—inheriting all the same challenges [38, 41, 198]. Moreover, even for problems involving only boundary terms, BEM must discretize the boundary geometry, leading to spatial aliasing in the boundary data (Fig. 8.15). Unlike FEM/meshless FEM, BEM must solve a globally coupled *dense* system of equations, demanding special techniques like *hierarchical matrix approximation* [93] to obtain reasonable performance. And unlike Monte Carlo, BEM does not allow for progressive or output-sensitive evaluation, as u and $\partial u/\partial n$ must always be determined on the entire boundary.

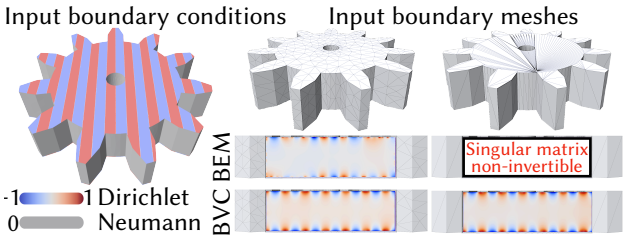


Figure 8.15: Even for problems with relatively simple boundary conditions and no source term (left), finite element technology like BEM suffers from large global errors in the PDE solution without significant mesh refinement, due to local aliasing of boundary data. It can also fail completely on domains with irregular elements. In contrast, Monte Carlo methods solve PDEs without any aliasing artifacts irrespective of tesselation quality, as they decouple the problem inputs from the boundary representation.

8.2.2 Finite Difference Methods

The main conceptual difference between finite difference and finite element methods is that degrees of freedom now represent point samples of the unknown function at nodal points, rather than coefficients in a finite basis. Derivatives are likewise evaluated via Taylor series approximation (*e.g.*, using finite difference formulas), rather than by taking derivatives of basis functions.

On the one hand, FD schemes are attractive due to the simplicity of implementation on a regular grid. However, a major challenge is spatial adaptivity: for many elliptic PDEs, refining the whole domain (with cubic growth in the number of grid cells) is overkill. Hierarchical structures like *octrees* can be used to adaptively refine solutions [78, 153], yet come with their own challenges (*e.g.*, less coherent memory access, and increased complexity of implementation). Enforcement of boundary conditions may also not be straightforward, since cell boundaries are typically axis aligned [33]. FD is also less than ideal for PDEs with nonuniform coefficients, where the uniformity of grid cells can lead to significant numerical diffusion [250], spurious negative values [230], and locking/stagnation [10].

On the whole, finite differences suffer from the same basic challenges as finite element methods: one must spatially discretize the domain, boundary conditions, source term, and coefficient functions, leading to either aliasing or oversampling. Moreover, one must solve a globally coupled system of equations over the entire domain, rather than concentrating computational effort only at points or regions of interest (as with Monte Carlo).

Material Point Methods. *Material point methods* [118], such as PIC [96], FLIP [29, 284], APIC [117], and MPM [243] are popular for time-dependent computational mechanics problems in-

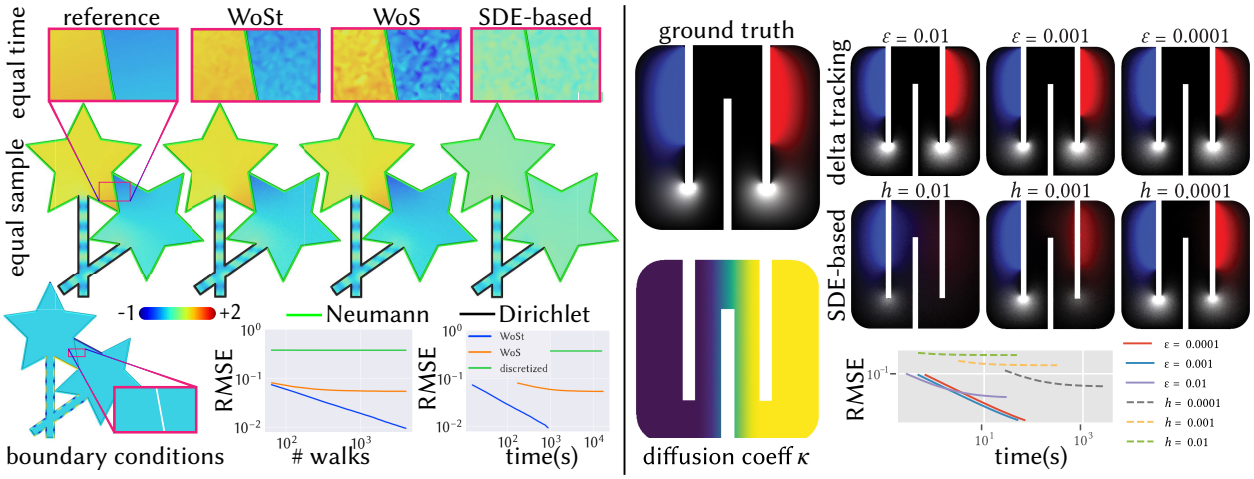


Figure 8.16: Left: for equal number of walks, WoSt is significantly more efficient than both WoS with discretized boundary reflections (Sec. 3.2.6) and SDE methods (Sec. 2.3.1). Right: our delta tracking WoS estimator correctly resolves the boundary conditions for any value of ε; shrinking the ε-shell reduces bias in the solution in a predicable manner, with little impact on performance. Solving the same PDE with an SDE estimator (Eq. 2.32) eventually resolves the boundary conditions with a finer step-size, but at the detriment of runtime performance. All timings were taken on an 8 core M1 MacBook Pro.

volving large-scale deformation (fluids, plasticity, *etc.*). These methods are also sometimes referred to as "meshless", but they are not (in general) meshless FEM schemes, as defined in Sec. 8.2.1. Rather, these methods use particles to approximate advection, and a background grid to solve elliptic problems (such as pressure projection in fluids). Critically, for the problems we consider here (time-*independent* PDEs), there is no advection component, and MPM reduces to simply solving elliptic equations on a grid—with the same trade-offs discussed above.

8.3 Comparison With Alternative Grid-Free Monte Carlo Solvers

Not all PDE solvers need to discretize space—the notable exception are Monte Carlo methods based on continuous random processes like Brownian motion. The stochastic approach centers on the simulation of random walks that in aggregate solve a large class of elliptic and parabolic PDEs [191]. Pointwise evaluation of PDE solutions has allowed this formulation to find extensive use in scientific disciplines such as mathematical finance [24, 43, 163, 164], computational physics and chemistry [80, 87, 160, 161] and optimal control [124, 127] (albeit on simple geometric domains).

8.3.1 Discretized Random Walks

As discussed in Sec. 2.3.1 & 2.3.2, the standard Monte Carlo approach simulates random walks with explicit time stepping, akin to ray marching in rendering. These integration schemes for stochastic differential equations [101, 172] are generally not well-suited for solving PDEs in complex domains, as a fixed time step either means slow runtimes due to a large number of steps inside the domain, or large bias in the estimated solution from discretizing walks.

As shown in Fig. 8.16 (*left*), WoSt is both significantly faster and less biased for mixed boundary-value problems with comparable parameter settings: epsilon value $\varepsilon = 0.001$ for WoSt, reflection offset $\zeta = 0.01$ for WoS, and step size $h = 0.0001$ for the SDE-based method. As discussed in Sec. 3.2.6, WoS with boundary reflections suffers from bias buildup due to long walks

that stick to the Neumann boundary. Methods based on reflecting SDEs perform even worse [42], as they incur bias not only on the boundary, but also in the interior. In contrast, the star-shaped regions used by WoSt enable one to take large steps without incurring significant bias.

Similar to ray marching [133], discretized walks still exhibit fairly predictable variance, as long as some bias in the solution is tolerable. Fig. 8.16 (*right*) shows that bias is exacerbated with SDE-based methods for variable-coefficient problems, as the diffusion and drift coefficients in Eq. 2.23 implicitly modify the ideal step size. In contrast, the ε-shell in WoS incurs only minuscule bias at the very end of a walk, leading to far less error overall.

8.3.2 Continuous Random Walks

A few grid-free Monte Carlo methods do not require any spatial or temporal discretization to simulate random walks. The chief example is WoS, which uses closed-form distributions to exactly model large steps of Brownian motion. Variants include walk on rectangles (WoR) [49] and walk on boundary (WoB) [217]. In the case of simple polygonal domains, WoR has been shown to converge faster to the boundary than WoS and without any bias, but it currently lacks accelerated geometric queries for finding the largest rectangle and has not been evaluated on complex geometric domains. WoB solves Laplacian BVPs with Dirichlet, Neumann and Robin boundary conditions by recursively evaluating single and double layer potentials using rays that reflect off the boundary. Unlike WoS and WoSt, WoB does not require any distance queries, but as we describe below, it currently suffers from very high variance and bias in non-convex domains.

The Walk On Boundary Method. Similar to WoSt, WoB uses direction sampling to determine the next walk location x_{k+1}, but uses the entire boundary $\partial\Omega$ as its sampling domain. This means that it must estimate the solution at all ray intersections with $\partial\Omega$, as each intersected point contributes to the solution estimate at x_k (Fig. 4.3, *left*). To avoid a branching walk that increases exponentially in size, WoB instead uses just a single randomly selected intersection. As shown in Fig. 8.17 & 8.18, this results in extremely high variance even in domains that are mostly convex, as the recursive solution estimate must be multiplied by the number of intersections to ensure the expected contribution from each intersection is correctly accounted for. Moreover, the Poisson kernel alternates sign between consecutive intersections in the WoB estimator [242, Sec. 4.1.1], which further results in unstable estimates due to cancellation [125, Ch. 4]. Sugimoto et al. [242, Sec. 4.1.2] therefore propose truncating walk length to reduce variance, but this approach introduces significant bias in non-convex domains (see Fig. 8.18, *fist column*). In contrast, WoSt has *much* more manageable variance and bias, as it only ever intersects ∂St once to select x_{k+1}, and uses an ε-shell that requires little-to-no hand-tuning.

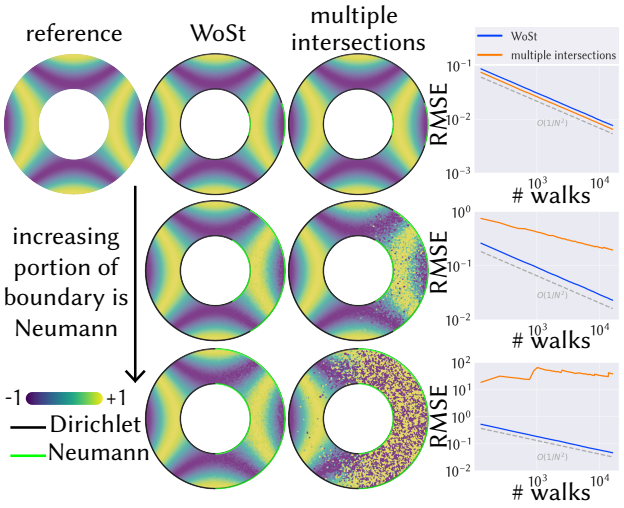
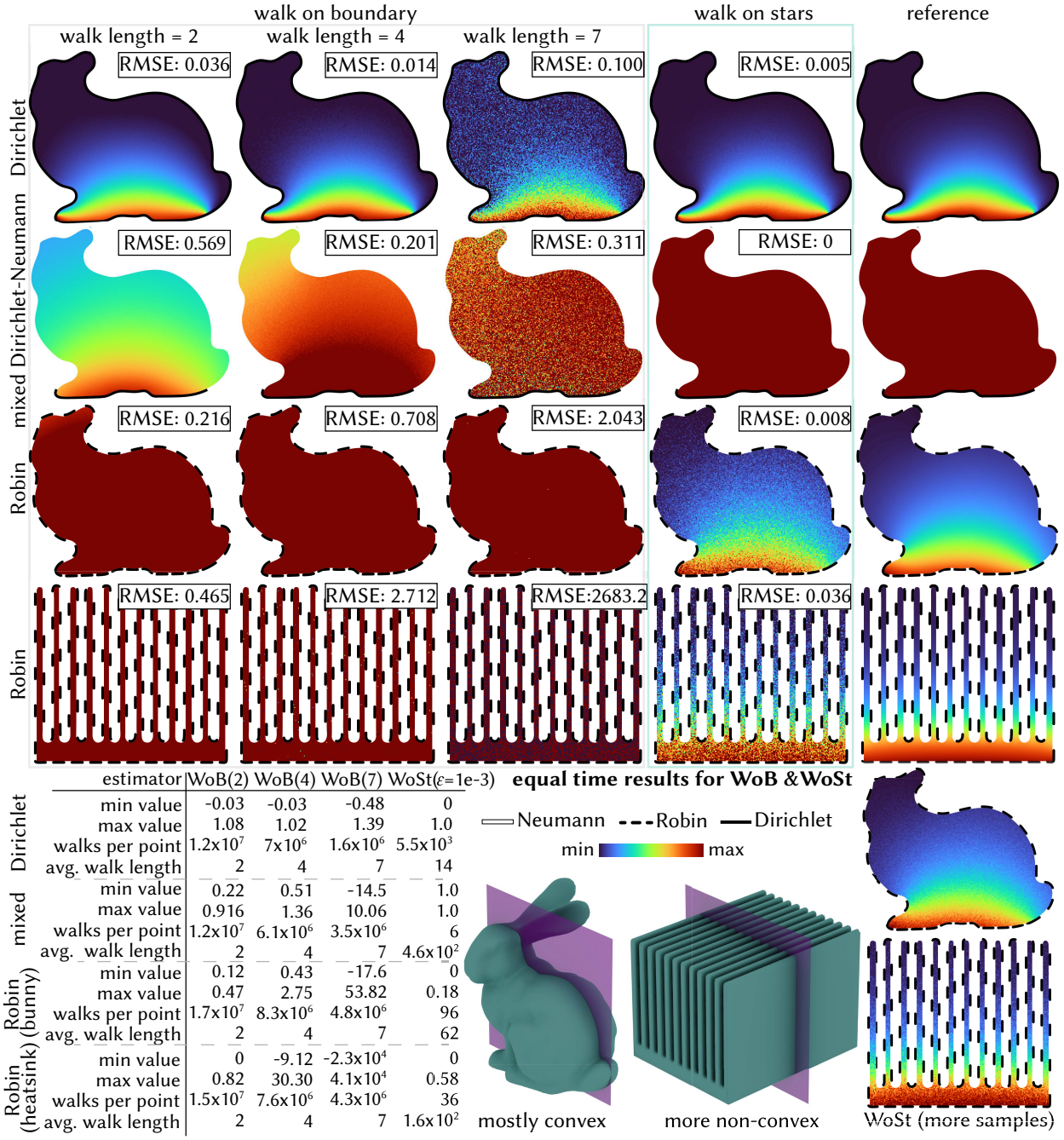


Figure 8.17: Here we solve for a known reference function, using its normal derivatives to specify Neumann conditions on an increasingly large part of the boundary. WoSt exhibits the expected Monte Carlo convergence rate, whereas estimators based on multiple ray intersections (Sec. 4.2.2), like walk on boundary, quickly blow up.



	estimator	WoB(2)	WoB(4)	WoB(7)	WoSt(ε=1e-3)
Dirichlet	min value	-0.03	-0.03	-0.48	0
	max value	1.08	1.02	1.39	1.0
	walks per point	$1.2\text{x}10^7$	$7\text{x}10^6$	$1.6\text{x}10^6$	$5.5\text{x}10^3$
	avg. walk length	2	4	7	14
mixed	min value	0.22	0.51	-14.5	1.0
	max value	0.916	1.36	10.06	1.0
	walks per point	$1.2\text{x}10^7$	$6.1\text{x}10^6$	$3.5\text{x}10^6$	6
	avg. walk length	2	4	7	$4.6\text{x}10^2$
Robin (bunny)	min value	0.12	0.43	-17.6	0
	max value	0.47	2.75	53.82	0.18
	walks per point	$1.7\text{x}10^7$	$8.3\text{x}10^6$	$4.8\text{x}10^6$	96
	avg. walk length	2	4	7	62
Robin (heatsink)	min value	0	-9.12	$-2.3\text{x}10^4$	0
	max value	0.82	30.30	$4.1\text{x}10^4$	0.58
	walks per point	$1.5\text{x}10^7$	$7.6\text{x}10^6$	$4.3\text{x}10^6$	36
	avg. walk length	2	4	7	$1.6\text{x}10^2$

Figure 8.18: While the walk on boundary method suffers from an extreme bias-variance tradeoff, WoSt demonstrates reliable Monte Carlo convergence with increasing sample count (i.e., *number of walks) for any combination of Dirichlet, Neumann and Robin boundary conditions. First three rows: Even in mostly convex domains with simple boundary conditions, WoB has noticeable bias in its solution estimate when walk length is truncated too aggressively (first column). Otherwise, WoB experiences an exponential increase in variance with longer walk lengths (second and third column), and requires an enormous number of samples to suppress error. RMSE remains high even for problems with a constant solution, whereas WoSt has no estimation error in this case as walk throughput is always bounded between 0 and 1 (second row). Fourth row: For more non-convex domains, variance in the solution estimate explodes with WoB even for a truncated walk length of 2—in contrast, estimation with WoSt is equally stable in both convex and non-convex domains.*

WoB has even more trouble with Robin problems (Fig. 8.18, *third and fourth rows*), as it has no mechanism to deal with the reflectance term $\rho_\mu = 1 - \mu G/P$ in its integral expressions (*e.g.*, via importance sampling). As a result, walk throughout typically grows even faster with Robin conditions, especially as μ is increased—here, standard variance reduction techniques from rendering cannot help bound throughput. Our WoSt estimator, on the other hand, ensures ρ_μ remains bounded between 0 and 1 on $\partial\mathrm{St}$. Therefore, we do not truncate walk length to a predefined value like WoB, and instead use Russian roulette to terminate walks *without any bias*.

Fig. 8.18 provides equal-time comparisons between WoB and WoSt for a Laplace equation, where we run the reference WoB implementation from Sugimoto et al. [242] on an Nvidia RTX 3090 GPU, and use a 12 core CPU machine for WoSt. WoSt demonstrates stable convergence with little bias for Dirichlet, Neumann and Robin boundary conditions. Though WoSt takes more steps on average per walk than WoB (see table on bottom left), the relative mean-squared error values illustrate that WoB is extremely sensitive to the choice of walk length, and requires an enormous number of samples to converge, even to a solution with large bias. This suggests that the example problems we used in Sec. 8.1 to stress-test WoSt are essentially intractable for WoB, due to the significant nonconvexity and complexity of their geometry.

Chapter 9

Future Directions

The goal of this work was to set the groundwork for Monte Carlo Geometry Processing by building on Muller [174]'s walk on spheres algorithm and fleshing out practical estimators for some of the most basic, yet fundamental partial differential equations in scientific, geometric and visual computing. In particular, we leveraged inherent properties of Brownian motion and drew on insights from rendering to consolidate the treatment of general first-order boundary conditions with WoSt (Ch. 4), provide the first estimators for fairly general continuously-varying coefficients (Ch. 5), and develop a number of variance reduction strategies for both the solution and its derivatives (Ch. 7). The key strength of our approach is its ability to function reliably in complex geometric domains without any preprocessing or volumetric meshing (Ch. 8); error decreases predictably with more samples and essentially no parameter tuning (Ch. 6).

As we discuss below, clearly a great deal more remains be done to enhance the efficiency of our solvers (Sec. 9.1), and broaden their applicability to a wider array of problems (Sec. 9.2). Yet, even within the class of PDEs presented here, geometric scalability will likely pay dividends in well-chosen scientific and engineering contexts—just as it has for Monte Carlo simulation of light transport. In general, the very different capabilities of grid-free Monte Carlo methods for solving PDEs are still largely unexplored in scientific, geometric and visual computing, with many unique benefits and attractive use cases yet to be discovered.

9.1 Enhancing Solver Efficiency

Though our estimators exhibit a much more favorable speed-bias tradeoff compared to other random walk methods (Sec. 8.3), they still require highly redundant computation compared to grid-based techniques, and can suffer from long walk lengths and noise. For instance, Fig. 9.1 replicates a classic "key-hole problem" using both 2D path tracing (assuming perfectly diffuse reflections) and WoSt (with a Neumann-dominated boundary). In both cases, several independent walks must make their way through the same narrow gap in the domain (with average walk length greater than 200), highlighting a general challenge faced by forward random walk estimators. Likewise, as in rendering, PDE coefficients with large spatial variation can result in small step-sizes and hence increased runtimes and variance—unlike rendering, large variations in $\sigma'(x)$ (Eq. 5.3) typically stem from *derivatives* of the diffusion and drift coefficients, which make it difficult to deal with very sharp changes in material density. Here we outline techniques that might broadly help improve efficiency by either taking less time to produce the same variance, or producing less variance in the same amount of time.

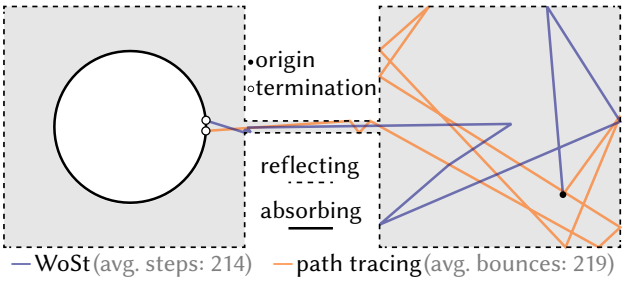


Figure 9.1: Random walk methods such as WoSt and path tracing require many steps or bounces to make their way through a key hole.

Geometric Queries. The dominant cost of our estimators is evaluating geometric queries (Sec. 6.1–6.2). While ray-intersections are well-optimized [259, 260], distance queries typically face bigger workload imbalance. For instance, depth first traversal provides good pruning near the domain boundary with CPQs, but becomes less effective for query points near the medial axis of the domain. This is because depth first traversal undergoes wasteful backtracking in regions equally close to different parts of the boundary. Further research and engineering effort is therefore required to perform millions of such queries in a cache-friendly fashion in parallel, while minimizing thread divergence amongst independent random walks with different walk lengths—possibly via a wavefront (rather than a megakernel) implementation on the GPU [142, 173].

Basic opportunities for accelerating CPQs and CSPQs include, *e.g.*, better tree construction heuristics [39, Sec. 4.4], use of oriented bounding volumes, and intelligent caching of silhouette edges. One might also extend silhouette queries to (neural) implicit surfaces by building on recent range analysis techniques [231]. Likewise, point sampling queries (Sec. 6.2.4) could be optimized by using accelerations developed for *many-lights sampling* [39, Sec. 5.4] to ensure that Neumann and Robin boundary samples lie inside star-shaped regions.

Next-Event, Path-Space And Hybrid Estimators. Random walk estimators may take many steps before obtaining a Dirichlet contribution, resulting in high computation time without a commensurate decrease in variance. This situation directly parallels rendering algorithms that produce long light paths in scenes with primarily non-absorbing or highly-scattering materials.

In rendering, *next-event estimation* reduces noise by adding a direct illumination contribution at each bounce [40, 268]. One could likewise try adding a Dirichlet contribution at each step of WoSt, by allowing subdomains to contain a subset of the Dirichlet boundary. Such a scheme would simply need some way to estimate $\partial u/\partial n$ at Dirichlet points (Eq. 4.1). More generally, a *path-space formulation* [203, Ch. 14] of the BIE may lead to estimators that make more global sampling decisions by connecting arbitrary evaluation points with difficult-to-sample boundary and source data (*e.g.*, via Markov chain Monte Carlo [131, 251] or path guiding [100, 175, 176, 177]). Moreover, the performance of our variable coefficient PDE estimators can likely be improved by adapting further techniques from volume rendering, such as local or progressive bounds on coefficient functions [169, 246, 274], and adaptive weight windows (as discussed in Sec. 7.7). It seems wise to also consider hybrid strategies that combine Monte Carlo estimation with information sharing, perhaps by way of temporal difference methods in reinforcement learning [245, Ch. 6].

Sample Reuse. When problems require dense evaluation, sample reuse strategies like BVC greatly reduce redundant computation by performing random walks only from the boundary (Sec. 7.5). We can improve BVC in a number of ways: principled and unbiased pointwise estimation of $\partial u/\partial n$ on the Dirichlet boundary should reduce global error in the estimated solution noticeably. Though sharing samples necessitates a lack of importance sampling, we can take further inspiration from VPL methods [46, Sec. 5] to suppress artifacts near the boundary, especially for gradients. We can also asymptotically reduce the quadratic complexity of evaluating the BIE by adapting clustering techniques such as Barnes-Hut and lightcuts [88, 152, 202, 264, 275].

Looking ahead, we can develop domain decomposition strategies [34] to more effectively handle domains with, *e.g.*, thin features where pointwise estimators struggle, as BVC can evaluate PDEs locally inside arbitrary bounding regions. BVC might even help speedup pointwise estimators by terminating walks early in regions where samples have previously been cached. More broadly, we can look for ways to unify boundary and mean value caching [11, 166], bidirectional random walks [167, 206] and neural caches [151] into a single caching framework for greater vari-

ance reduction, without compromising on the scalability, progressivity, and output sensitivity of the underlying estimators. Other reuse schemes from rendering for dynamic geometry such as ReSTIR [23, 193] and recursive control variates [186] provide similar opportunities.

Concave Reflecting Boundaries. To take large steps near silhouette points (Fig. 4.5), we can replace the fixed parameter ε with an adaptive radius based on local curvature estimates [205]. Alternatively, one might apply *multi-level Monte Carlo* [79] to reduce bias by aggregating estimates obtained via progressively smaller values of ε. Even more generally, we can look for subdomains other than star-shaped regions to take larger steps while keeping throughput bounded.

Denoising and Geometric Prefiltering. Since elliptic PDEs have very regular solutions, high-frequency noise in WoSt estimates can be nicely mitigated via denoising (Fig. 7.6); here again methods from rendering provide a wealth of opportunities [137, 189, 190, 226, 288]. Another interesting challenge is how to detect—or even define—silhouettes for geometry with intricate microstructures [184], perhaps through some form of geometric prefiltering [271]. In particular, the SNCH (Sec. 6.2.1) itself provides a form of prefiltering: nodes higher than the leaves can provide conservative or approximate bounds on $d_{\text{silhouette}}$ that effectively amount to smoothing out fine-scale geometry (at the cost of bias). Likewise, SNCH nodes could be built by sampling a distribution (*e.g.*, a microflake model [99]) rather than bounding explicit geometry.

9.2 Broadening Solver Applicability

Extending our Robin estimators to support both positive and negative coefficients should enable us to solve not just variable coefficient PDEs with reflecting boundary conditions (as discussed in Sec. 5.3), but also exterior problems via the method of Nabizadeh et al. [179]—here the *Kelvin transform* converts Neumann conditions into Robin. With Robin conditions, our WoSt estimator can be used alongside a ray tracer to more accurately model physics that couples conduction, convection and radiative transfer [14]. The next logical extension for the variable coefficient estimators from Ch. 5 is to generalize the isotropic diffusion coefficient $\kappa(x)$ to be matrix valued, possibly through the *Lamperti transform* [170]: such anisotropic coefficients currently present significant challenges for conventional PDE solvers due to the need for anisotropic mesh generation [10, 230, 250], which is an even harder problem than isotropic meshing [214]; this extension will also enable simulation of random walks on more general curved surfaces than those in Fig. 8.8.

Though we have developed our WoS and WoSt estimators in the context of linear elliptic PDEs like the Poisson equation, the basic algorithmic strategy should apply more broadly: fundamentally, these estimators depend on the structure of the BIE, and BIE formulations are readily available for a variety of other important PDEs in scientific computing, including the heat [95], Helmholtz [18, 106, Ch. 3] and biharmonic equations [111], as well as linear elasticity [106, Ch. 4]. Stochastic integral formulations are known for Navier-Stokes [31, 211], while other nonlinear PDEs can be handled by, *e.g.*, simulating *branching diffusion* [28], or by applying *forward-backward stochastic differential equations* [196]. We did not touch on the possibility of solving higher dimensional problems in this work, or PDEs with unknown, random variables (as often arise when working with measured data), which also seem quite natural in the Monte Carlo setting.

Lastly, as with differentiable rendering [83, 150, 187, 278], an exciting direction of future work, and a likely source of many applications, is to develop differentiable variants of WoS that optimize parameterized descriptions of geometry, boundary conditions and coefficients for inverse tasks like impedance tomography [272] and thermally aware circuit board design [32].

Bibliography

[1] Assyr Abdulle, E Weinan, Björn Engquist, and Eric Vanden-Eijnden. The heterogeneous multiscale method. *Acta Numerica*, 21:1–87, 2012.

[2] William Abikoff. The uniformization theorem. *Amer. Math. Monthly*, 88(8), 1981.

[3] S. Alanko and M. Avellaneda. Reducing variance in the numerical solution of BSDEs. *Comptes Rendus Mathematique*, 351(3-4):135–138, 2013.

[4] Robert Anderson, Julian Andrej, Andrew Barker, et al. MFEM: A modular finite element methods library. *Computers & Mathematics with Applications*, 81, 2021.

[5] J. Arvo. Stratified sampling of 2-manifolds. *SIGGRAPH Course Notes*, 29(2), 2001.

[6] James Arvo. The role of functional analysis in global illumination. In *Rendering Techniques' 95: Proceedings of the Eurographics Workshop in Dublin, Ireland, June 12–14, 1995 6*, pages 115–126. Springer, 1995.

[7] James Richard Arvo. *Analytic methods for simulated light transport*. PhD thesis, Yale University, 1995.

[8] M. Attene, M. Campen, and L. Kobbelt. Polygon mesh repairing: An application perspective. *ACM Computing Surveys (CSUR)*, 45(2):15, 2013.

[9] Sheldon Axler, Paul Bourdon, and Wade Ramey. *Harmonic Function Theory*. Springer New York, NY, 2013. ISBN 978-0-387-95218-5. doi: https://doi.org/10.1007/978-1-4757-8137-3.

[10] Ivo Babuška and Manil Suri. On locking and robustness in the finite element method. *SIAM Journal on Numerical Analysis*, 29(5):1261–1293, 1992.

[11] Ghada Bakbouk and Pieter Peers. Mean value caching for walk on spheres. In Tobias Ritschel and Andrea Weidlich, editors, *Eurographics Symposium on Rendering*. The Eurographics Association, 2023. ISBN 978-3-03868-229-5. doi: 10.2312/sr.20231120.

[12] G. Barill, N. G Dickson, R. Schmidt, D. Levin, and A. Jacobson. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics*, 37(4):1–12, 2018.

[13] Bruce A Barnes. Spectral properties of linear Volterra operators. *Journal of Operator Theory*, pages 365–382, 1990.

[14] Mégane Bati, Stéphane Blanco, Christophe Coustet, Vincent Eymet, Vincent Forest, Richard Fournier, Jacques Gautrais, Nicolas Mellado, Mathias Paulin, and Benjamin Piaud. Coupling conduction, convection and radiative transfer in a single path-space: Application to infrared rendering. *ACM Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301. doi: 10.1145/3592121. URL https://doi.org/10.1145/3592121.

[15] Victor Bayona, Natasha Flyer, Bengt Fornberg, and Gregory A Barnett. On the role of polynomials in RBF-FD approximations: Ii. numerical solution of elliptic PDEs. *Journal of Computational Physics*, 332:257–273, 2017.

[16] Victor Bayona, Natasha Flyer, and Bengt Fornberg. On the role of polynomials in RBF-FD approximations: Iii. behavior near domain boundaries. *Journal of Computational Physics*, 380:378–399, 2019.

[17] D. Bell. *The Malliavin Calculus*. Courier Corporation, 2012.

[18] Francisco Bernal, Xingyuan Chen, and Goncalo dos Reis. An iterative method for helmholtz boundary value problems arising in wave propagation. *arXiv preprint arXiv:2308.11469*, 2023.

[19] G. Bernstein and D. Fussell. Fast, exact, linear booleans. In *Computer Graphics Forum*, volume 28, pages 1269–1278. Wiley Online Library, 2009.

[20] H. Bhatia, G. Norgard, V. Pascucci, and P. Bremer. The Helmholtz-Hodge decomposition—a survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(08), 2013. ISSN 1941-0506.

[21] H. Bhatia, V. Pascucci, and P. Bremer. The natural Helmholtz-Hodge decomposition for open-boundary flow analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20 (11), 2014.

[22] Ilia Binder and Mark Braverman. The rate of convergence of the walk on spheres algorithm. *Geometric and Functional Analysis*, 22(3):558–587, 2012.

[23] 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 Trans. Graph.*, 39(4), aug 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392481. URL https://doi.org/10.1145/3386569.3392481.

[24] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–54, 1973. URL https://EconPapers.repec.org/RePEc:ucp:jpolec:v:81:y:1973:i:3:p:637-54.

[25] Thomas E Booth. Monte Carlo variance comparison for expected-value versus sampled splitting. *Nuclear Science and Engineering*, 89(4):305–309, 1985.

[26] Thomas E Booth and John S Hendricks. Importance estimation in forward Monte Carlo calculations. *Nuclear Technology-Fusion*, 5(1):90–100, 1984.

[27] Andrei N Borodin and Paavo Salminen. *Handbook of Brownian motion-facts and formulae*. Springer Science & Business Media, 2015.

[28] Mireille Bossy, Nicolas Champagnat, Hélène Leman, Sylvain Maire, Laurent Violeau, and Mariette Yvinec. Monte Carlo methods for linear and non-linear Poisson-Boltzmann equation. *ESAIM: Proceedings and Surveys*, 48:420–446, 2015.

[29] Jeremiah U Brackbill and Hans M Ruppel. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows. *J. Comp. Phys.*, 65(2):314–343, 1986.

[30] R. Bridson. Fast Poisson disk sampling in arbitrary dimensions. In *SIGGRAPH sketches*, page 22, 2007.

[31] 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. *Proc. Edinburgh Math. Soc.*, 48(2):295–336, 2005.

[32] Cadence. Using a thermal FEA solver for heat management in your PCB, 2024. URL https://resources.pcb.cadence.com/blog/2020-using-a-thermal-fea-solver-for-heat-management-in-your-pcb.

[33] DM Causon and CG Mingham. *Introductory finite difference methods for PDEs*. Bookboon, 2010.

[34] Tony F Chan and Tarek P Mathew. Domain decomposition algorithms. *Acta numerica*, 3: 61–143, 1994.

[35] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Dover Books on Advanced Mathematics. Dover Publications, NY, 1960. ISBN 0-486-60590-6.

[36] Peter Yichen Chen, Jonathan David Blutinger, Yorán Meijers, Changxi Zheng, Eitan Grinspun, and Hod Lipson. Visual modeling of laser-induced dough browning. *Journal of Food Engineering*, 243:9–21, 2019.

[37] X. Chen, Y. Zhou, Z. Shu, H. Su, and J. Paul. Improved algebraic algorithm on point projection for b´ eziercurves. In *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, pages 158–163. IEEE, 2007.

[38] Chris J Coleman, David L Tullock, and Nhan Phan-Thien. An effective boundary element method for inhomogeneous PDEs. *J. App. Math. Phys. (ZAMP)*, 42(5), 1991.

[39] Alejandro Conty Estevez and Christopher Kulla. Importance sampling of many lights with adaptive tree splitting. *Proc. ACM Comput. Graph. Interact. Tech.*, 1(2), aug 2018. doi: 10.1145/3233305. URL https://doi.org/10.1145/3233305.

[40] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proc. SIGGRAPH*, page 137–145, 1984.

[41] Martin Costabel. Principles of boundary element methods. *Computer Physics Reports*, 6(1-6): 243–274, 1987.

[42] Cristina Costantini, Barbara Pacchiarotti, and Flavio Sartoretto. Numerical approximation for functionals of reflecting diffusion processes. *SIAM Journal on Applied Mathematics*, 58(1): 73–102, 1998.

[43] John C. Cox, Jonathan E. Ingersoll, and Stephen A. Ross. A theory of the term structure of interest rates. *Econometrica*, 53(2):385–407, 1985. ISSN 00129682, 14680262.

[44] S. N. Cramer. Application of the fictitious scattering radiation transport model for deep-penetration Monte Carlo calculations. *Nuclear Science and Engineering*, 65(2):237–253, 1978. doi: 10/gfzq74.

[45] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, 32 (5):1–11, 2013.

[46] Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. Scalable realistic rendering with many-light methods. In *Computer Graphics Forum*, volume 33, pages 88–104. Wiley Online Library, 2014.

[47] Auguste De Lambilly, Gabriel Benedetti, Nour Rizk, Chen Hanqi, Siyuan Huang, Junnan Qiu, David Louapre, Raphael Granier De Cassagnac, and Damien Rohmer. Heat simulation on meshless crafted-made shapes. In *Proceedings of the 16th ACM SIGGRAPH Conference on Motion, Interaction and Games*, MIG '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400703935. doi: 10.1145/3623264.3624457. URL https://doi.org/10.1145/3623264.3624457.

[48] M. Deaconu, S. Herrmann, and S. Maire. The walk on moving spheres: A new tool for simulating brownian motion's exit time from a domain. *Mathematics and Computers in Simulation*, 135:28–38, 2017. ISSN 0378-4754. doi: https://doi.org/10.1016/j.matcom.2015.07.004. URL https://www.sciencedirect.com/science/article/pii/S0378475415001366. Special Issue: 9th IMACS Seminar on Monte Carlo Methods.

[49] Madalina Deaconu and Antoine Lejay. Simulation of diffusions by means of importance sampling paradigm. *The Annals of Applied Probability*, 20(4):1389–1424, 2010. ISSN 10505164. URL http://www.jstor.org/stable/20744098.

[50] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. *Acm siggraph computer graphics*, 22(4):21–30, 1988.

[51] J. M. Delaurentis and L. A. Romero. A Monte Carlo method for Poisson's equation. *Journal of Computational Physics*, 90(1):123–140, 1990. doi: 10.1016/0021-9991(90)90199-B.

[52] Mathieu Desbrun, Roger D Donaldson, and Houman Owhadi. Modeling across scales: Discrete geometric structures in homogenization and inverse homogenization. *Multiscale Analysis and Nonlinear Dynamics*, pages 19–64, 2013.

[53] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 275–286. ACM, 1998.

[54] L. Devroye. Sample-based non-uniform random variate generation. In *Proceedings of the 18th conference on Winter simulation*, pages 260–265. ACM, 1986.

[55] Dean G Duffy. *Green's functions with applications*. Chapman and Hall/CRC, 2015.

[56] Louis J Durlofsky. Numerical calculation of equivalent grid block permeability tensors for heterogeneous porous media. *Water resources research*, 27(5), 1991.

[57] Bruce B Dykaar and Peter K Kitanidis. Determination of the effective hydraulic conductivity for heterogeneous porous media. *Water Res. R.*, 28(4), 1992.

[58] E. Dyllong and W. Luther. Distance calculation between a point and a NURBS surface. Technical report, Duisburg University, 2000.

[59] G. Dziuk. Finite elements for the beltrami operator on arbitrary surfaces. In *Partial differential equations and calculus of variations*, pages 142–155. Springer, 1988.

[60] Yalchin Efendiev and Thomas Y Hou. *Multiscale finite element methods: theory and applications*, volume 4. 2009.

[61] B. Elepov and G. Mikhailov. Solution of the Dirichlet problem for the equation $\delta u - cu = -q$ by a model of "walks on spheres". *USSR Computational Mathematics and Mathematical Physics*, 9(3):194–204, 1969.

[62] Christer Ericson. *Real-time collision detection*. Crc Press, 2004.

[63] Sergey M Ermakov and Alexander S Sipin. The "walk in hemispheres" process and its applications to solving boundary value problems. *Vestnik St. Petersburg University: Mathematics*, 42:155–163, 2009. ISSN 1934-7855.

[64] Lawrence C Evans. *Partial differential equations*, volume 19. Rhode Island, USA, 1998.

[65] SP Eveson. Norms of iterates of Volterra operators on l 2. *Journal of Operator Theory*, pages 369–386, 2003.

[66] Florian Fahrenberger, Zhenli Xu, and Christian Holm. Simulation of electric double layers around charged colloids in aqueous solution of variable permittivity. *J. Chem. Phys.*, 141(6), 2014.

[67] Marcel Filoche and Bernard Sapoval. Can one hear the shape of an electrode? ii. theoretical study of the Laplacian transfer. *The European Physical Journal B-Condensed Matter and Complex Systems*, 9:755–763, 1999.

[68] J. A Fleck and E. H Canfield. A random walk procedure for improving the computational efficiency of the implicit Monte Carlo method for nonlinear radiation transport. *Journal of Computational Physics*, 54(3):508–523, June 1984. ISSN 0021-9991.

[69] Natasha Flyer, Bengt Fornberg, Victor Bayona, and Gregory A Barnett. On the role of polynomials in RBF-FD approximations: I. interpolation and accuracy. *Journal of Computational Physics*, 321:21–38, 2016.

[70] Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. Production volume rendering: SIGGRAPH 2017 course. In *ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350143. doi: 10.1145/3084873.3084907. URL https://doi.org/10.1145/3084873.3084907.

[71] Eric Fournié, Jean-Michel Lasry, Jérôme Lebuchoux, Pierre-Louis Lions, and Nizar Touzi. Applications of Malliavin calculus to Monte Carlo methods in finance. *Finance and Stochastics*, 3(4):391–412, 1999.

[72] A. Friedman and K.S. Fu. *Stochastic Differential Equations and Applications*. 1975. ISBN 9780122682018.

[73] Thomas-Peter Fries, Hermann Matthies, et al. Classification and overview of meshfree methods. 2004.

[74] M. Galtier, S. Blanco, C. Caliot, C. Coustet, J. Dauchet, M. El Hafi, V. Eymet, R. Fournier, J. Gautrais, A. Khuong, B. Piaud, and G. Terrée. Integral formulation of null-collision Monte Carlo algorithms. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 125:57–68, August 2013. ISSN 0022-4073.

[75] Riccardo Gatto. The von Mises–Fisher distribution of the first exit point from the hypersphere of the drifted Brownian motion and the density of the first exit time. *Statistics & Probability Letters*, 83(7):1669–1676, 2013.

[76] Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, Adrien Herubel, Declan Russell, Frédéric Servant, and Marcos Fajardo. Arnold: A brute-force production path tracer. *ACM Trans. Graph.*, 37(3), aug 2018. ISSN 0730-0301. doi: 10.1145/3182160. URL https://doi.org/10.1145/3182160.

[77] Iliyan Georgiev, Zackary Misso, Toshiya Hachisuka, Derek Nowrouzezahrai, Jaroslav Křivánek, and Wojciech Jarosz. Integral formulations of volumetric transmittance. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6):154:1–154:17, November 2019. ISSN 0730-0301. doi: 10/dffn.

[78] Frederic Gibou, Ronald Fedkiw, and Stanley Osher. A review of level-set methods and some recent applications. *Journal of Computational Physics*, 353:82–109, 2018.

[79] Michael B Giles. Multilevel Monte Carlo methods. *Acta Numer.*, 24:259–328, 2015.

[80] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[81] James A. Given, Joseph B. Hubbard, and Jack F. Douglas. A first-passage algorithm for the hydrodynamic friction and diffusion-limited reaction rate of macromolecules. *The Journal of Chemical Physics*, 106(9):3761–3771, 03 1997. ISSN 0021-9606. doi: 10.1063/1.473428. URL https://doi.org/10.1063/1.473428.

[82] D. Givoli, L. Rivkin, and J. Keller. A finite element method for domains with corners. *International journal for numerical methods in engineering*, 35(6), 1992.

[83] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse volume rendering with material dictionaries. *ACM Trans. Graph.*, 32(6), nov 2013. ISSN 0730-0301. doi: 10.1145/2508363.2508377. URL https://doi.org/10.1145/2508363.2508377.

[84] K Gopalsamy and BD Aggarwala. Monte Carlo methods for some fourth order partial differential equations. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 50(12):759–767, 1970.

[85] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, page 213–222, New York, NY, USA, 1984. Association for Computing Machinery. ISBN 0897911385. doi: 10.1145/800031.808601. URL https://doi.org/10.1145/800031.808601.

[86] Denis S Grebenkov. Partially reflected Brownian motion: a stochastic approach to transport phenomena. *Focus on probability theory*, pages 135–169, 2006.

[87] Denis S Grebenkov. NMR survey of reflected Brownian motion. *Reviews of Modern Physics*, 79(3):1077, 2007.

[88] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[89] Jerry Jinfeng Guo and Elmar Eisemann. Geometric sample reweighting for Monte Carlo integration. In *Computer Graphics Forum*, volume 40, pages 109–119. Wiley Online Library, 2021.

[90] Karl Gustafson and Takehisa Abe. The third boundary condition—was it Robin's? *The Mathematical Intelligencer*, 20(1):63–71, 1998.

[91] Toshiya Hachisuka and Henrik Wann Jensen. Stochastic progressive photon mapping. *ACM Trans. Graph.*, 28(5):1–8, dec 2009. ISSN 0730-0301. doi: 10.1145/1618452.1618487. URL https://doi.org/10.1145/1618452.1618487.

[92] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. *ACM Trans. Graph.*, 27(5), dec 2008. ISSN 0730-0301. doi: 10.1145/1409060.1409083. URL https://doi.org/10.1145/1409060.1409083.

[93] Wolfgang Hackbusch. *Hierarchical matrices: algorithms and analysis*, volume 49. Springer, 2015.

[94] David W Hahn and M Necati Özisik. *Heat conduction*. John Wiley & Sons, 2012.

[95] A. Haji-Sheikh and E. M. Sparrow. The floating random walk and its application to Monte Carlo solutions of heat equations. *SIAM Journal on Applied Mathematics*, 14(2):370–389, 1966. ISSN 00361399. URL http://www.jstor.org/stable/2946271.

[96] Francis H Harlow and J Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8(12), 1965.

[97] J. Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[98] Stefan Heinrich and Peter Mathé. The Monte Carlo complexity of fredholm integral equations. *mathematics of computation*, 60(201):257–278, 1993.

[99] Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. The SGGX microflake distribution. *ACM Trans. Graph.*, 34(4), 2015.

[100] Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik P A Lensch, and Jaroslav Křivánek. Volume path guiding based on zero-variance random walk theory. *ACM Trans. Graph.*, 38(3), 2019.

[101] Desmond J. Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Review*, 43(3):525–546, 2001. doi: 10.1137/S0036144500378302. URL https://doi.org/10.1137/S0036144500378302.

[102] J Eduard Hoogenboom and Dávid Légrády. A critical review of the weight window generator in MCNP. In *Proceedings of Monte Carlo 2005 Topical Meeting, Chattanooga, TN on CD-ROM*, pages 17–21, 2005.

[103] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo. Tetrahedral meshing in the wild. *ACM Transactions on Graphics*, 37(4):60–1, 2018.

[104] Y. Hu, T. Schneider, X. Gao, Q. Zhou, A. Jacobson, D. Zorin, and D. Panozzo. Triwild: Robust triangulation with curve constraints. *ACM Transactions on Graphics*, 38(4):52, 2019.

[105] Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. Fast tetrahedral meshing in the wild. *ACM Trans. Graph.*, 39(4), aug 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392385. URL https://doi.org/10.1145/3386569.3392385.

[106] Peter Hunter and Andrew Pullan. FEM/BEM notes. *Department of Engineering Science, The University of Auckland, New Zeland*, 2001.

[107] Chi-Ok Hwang and Michael Mascagni. Analysis and comparison of green's function first-passage algorithms with "walk on spheres" algorithms. *Mathematics and Computers in Simulation*, 63(6):605–613, 2003. ISSN 0378-4754. doi: https://doi.org/10.1016/S0378-4754(03)00091-0. URL https://www.sciencedirect.com/science/article/pii/S0378475403000910.

[108] Chi-Ok Hwang and Michael Mascagni. Electrical capacitance of the unit cube. *Journal of Applied Physics*, 95(7):3798–3802, 03 2004. ISSN 0021-8979. doi: 10.1063/1.1664031. URL https://doi.org/10.1063/1.1664031.

[109] Chi-Ok Hwang, James A. Given, and Michael Mascagni. On the rapid estimation of permeability for porous media using Brownian motion paths. *Physics of Fluids*, 12(7):1699–1709, 07 2000. ISSN 1070-6631. doi: 10.1063/1.870420. URL https://doi.org/10.1063/1.870420.

[110] Chi-Ok Hwang, Sungpyo Hong, and Jinwoo Kim. Off-centered "walk-on-spheres" (WOS) algorithm. *Journal of Computational Physics*, 303:331–335, 2015. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2015.10.002. URL https://www.sciencedirect.com/science/article/pii/S0021999115006646.

[111] Derek B Ingham and Mark A Kelmanson. *Boundary integral equation analyses of singular, potential, and biharmonic problems*, volume 7. Springer Sci. Bus. Med., 2012.

[112] Intel. Open image denoise, 2019. https://openimagedenoise.github.io/.

[113] Kurt Jacobs. *Stochastic processes for physicists: understanding noisy systems*. Cambridge University Press, 2010.

[114] A. Jacobson, L. Kavan, and O. Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics*, 32(4), 2013.

[115] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques' 96: Proceedings of the Eurographics Workshop in Porto, Portugal, June 17–19, 1996 7*, pages 21–30. Springer, 1996.

[116] Henrik Wann Jensen, James Arvo, Marcos Fajardo, Patrick Hanrahan, D Mitchel, D Pharr, and Peter Shirley. State of the art in Monte Carlo ray tracing for realistic image synthesis. *SIGGRAPH 2001 Course Notes*, 2, 2001.

[117] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics*, 34(4):1–10, 2015.

[118] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, pages 1–52. 2016.

[119] David E Johnson and Elaine Cohen. Spatialized normal come hierarchies. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 129–134, 2001.

[120] Konrad Jörgens. *Linear integral operators*. Pitman, 1982.

[121] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, page 143–150, New York, NY, USA, 1986. Association for Computing Machinery. ISBN 0897911962. doi: 10.1145/15922.15902. URL https://doi.org/10.1145/15922.15902.

[122] S. Kakutani. Two-dimensional Brownian motion and harmonic functions. *Proceedings of the Imperial Academy*, 20(10):706–714, 1944.

[123] N. Kalantari, S. Bako, and P. Sen. A machine learning approach for filtering Monte Carlo noise. *ACM Transactions on Graphics*, 34(4):122–1, 2015.

[124] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[125] Malvin H Kalos and Paula A Whitlock. *Monte Carlo methods*. John Wiley & Sons, 2009. ISBN 9783527626212. doi: 10.1002/9783527626212.

[126] Ehsan Kamel and Ali Kazemian. BIM-integrated thermal analysis and building energy modeling in 3d-printed residential buildings. *Energy and Buildings*, 279:112670, 2023.

[127] Hilbert J Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. In *AIP conference proceedings*, volume 887. Amer. Inst. Phys., 2007.

[128] Tosio Kato. *Perturbation theory for linear operators*, volume 132. Springer Science & Business Media, 2013.

[129] Michael Kazhdan and Hugues Hoppe. Screened Poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[130] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[131] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A simple and robust mutation strategy for the Metropolis light transport algorithm. In *Computer Graphics Forum*, volume 21, pages 531–540. Wiley Online Library, 2002.

[132] Alexander Keller. Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 49–56, USA, 1997. ISBN 0897918967. doi: 10.1145/258734.258769. URL https://doi.org/10.1145/258734.258769.

[133] Markus Kettunen, Eugene D'Eon, Jacopo Pantaleoni, and Jan Novák. An unbiased ray-marching transmittance estimator. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459937. URL https://doi.org/10.1145/3450626.3459937.

[134] Peter E Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer Berlin, Heidelberg, 1992. ISBN 978-3-662-12616-5. doi: https://doi.org/10.1007/978-3-662-12616-5.

[135] Thomas Kollig and Alexander Keller. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 245–257, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-31186-7.

[136] D. Kopta, T. Ize, J. Spjut, E. Brunvand, A. Davis, and A. Kensler. Fast, effective BVH updates for animated scenes. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 197–204. ACM, 2012.

[137] Pawel Kozlowski and Tim Cheblokov. ReLAX: A denoiser tailored to work with the ReSTIR algorithm. *GPU Technology Conference*, 2021.

[138] Bastian Krayer and Stefan Müller. Hierarchical point distance fields. In *International Symposium on Visual Computing*, pages 435–446. Springer, 2021.

[139] Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. Spectral and decomposition tracking for rendering heterogeneous volumes. *ACM Transactions on Graphics (TOG)*, 36(4), July 2017. ISSN 0730-0301.

[140] Eric P Lafortune and Yves D Willems. Bi-directional path tracing. 1993.

[141] Eric P. Lafortune and Yves D. Willems. Rendering participating media with bidirectional path tracing. In *Rendering Techniques' 96: Proceedings of the Eurographics Workshop in Porto, Portugal, June 17–19, 1996 7*, pages 91–100, Vienna, June 1996. Springer. ISBN 978-3-211-82883-0.

[142] Samuli Laine, Tero Karras, and Timo Aila. Megakernels considered harmful: wavefront path tracing on GPUs. In *Proceedings of the 5th High-Performance Graphics Conference*, HPG '13, page 137–143, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321358. doi: 10.1145/2492045.2492060. URL https://doi.org/10.1145/2492045.2492060.

[143] Eric Larsen, Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes. Technical report, Technical Report TR99-018, Department of Computer Science, University of . . . , 1999.

[144] Greg Ward Larson and Rob Shakespeare. *Rendering with radiance: the art and science of lighting visualization*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998. ISBN 1558604995.

[145] J. Lawrence, S. Rusinkiewicz, and R. Ramamoorthi. Efficient BRDF importance sampling using a factored representation. In *ACM Transactions on Graphics*, volume 23. ACM, 2004.

[146] Y.L. Le Coz and R.B. Iverson. A stochastic algorithm for high speed capacitance extraction in integrated circuits. *Solid-State Electronics*, 35(7):1005–1012, 1992. ISSN 0038-1101. doi: https://doi.org/10.1016/0038-1101(92)90332-7. URL https://www.sciencedirect.com/science/article/pii/0038110192903327.

[147] Richard Lee and Carol O'Sullivan. Accelerated Light Propagation Through Participating Media. In H.-C. Hege, R. Machiraju, T. Moeller, and M. Sramek, editors, *Eurographics/IEEE VGTC Symposium on Volume Graphics*. The Eurographics Association, 2007. ISBN 978-3-905674-03-3. doi: 10.2312/VG/VG07/017-023.

[148] Antoine Lejay and Sylvain Maire. New Monte Carlo schemes for simulating diffusions in discontinuous media. *J. Comp. and Appl. Mathematics*, 245, 2013.

[149] S. Li and W. Liu. *Meshfree Particle Methods*. Springer Publishing Company, Incorporated, 2007. ISBN 3540222561.

[150] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans. Graph.*, 37(6), dec 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275109. URL https://doi.org/10.1145/3272127.3275109.

[151] Zilu Li, Gundao Yang, Xi Deng, Christopher De Sa, Bharat Hariharan, and Steve Marschner. Neural cache for Monte Carlo partial differential equation solver. *ACM Trans. Graph.*, 42(6), nov 2023. ISSN 0730-0301. doi: 10.1145/3610548.3618141. URL https://doi.org/10.1145/3610548.3618141.

[152] Daqi Lin and Cem Yuksel. Real-time stochastic lightcuts. *Proc. ACM Comput. Graph. Interact. Tech.*, 3(1), may 2020. doi: 10.1145/3384543. URL https://doi.org/10.1145/3384543.

[153] Frank Losasso, Ronald Fedkiw, and Stanley Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids*, 35(10):995–1010, 2006.

[154] V. L. Lukinov and G. A. Mikhailov. Probabilistic representation and Monte Carlo methods for the first boundary value problem for a polyharmonic equation. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 19(5):433–448, 2004. doi: doi:10.1515/1569398042395989. URL https://doi.org/10.1515/1569398042395989.

[155] Sylvain Maire and Giang Nguyen. Stochastic finite differences for elliptic diffusion equations in stratified domains. *Mathematics & Comp. in Simulation*, 121, 2016.

[156] Sylvain Maire and Etienne Tanré. Monte Carlo approximations of the Neumann problem. *Monte Carlo Methods and Applications*, 19(3):201–236, 2013.

[157] Nathan G March, Elliot J Carr, and Ian W Turner. Numerical investigation into coarse-scale models of diffusion in complex heterogeneous media. *Transport in Porous Media*, 139(3): 467–489, 2021.

[158] Zoë Marschner, Paul Zhang, David Palmer, and Justin Solomon. Sum-of-squares geometry processing. *ACM Transactions on Graphics (TOG)*, 40(6):1–13, 2021.

[159] M. Mascagni and C. Hwang. ϵ-shell error analysis for "walk on spheres" algorithms. *Mathematics and computers in simulation*, 63(2):93–104, 2003.

[160] Michael Mascagni and Nikolai A. Simonov. Monte Carlo methods for calculating some physical properties of large molecules. *SIAM Journal on Scientific Computing*, 26(1):339–357, 2004. doi: 10.1137/S1064827503422221. URL https://doi.org/10.1137/S1064827503422221.

[161] Michael Mascagni and Nikolai A Simonov. The random walk on the boundary method for calculating capacitance. *Journal of Computational Physics*, 195(2):465–473, 2004.

[162] Khamron Mekchay and Ricardo H Nochetto. Convergence of adaptive finite element methods for general second order linear elliptic PDEs. *SIAM Journal on Numerical Analysis*, 43 (5):1803–1827, 2005.

[163] Robert C Merton. Optimum consumption and portfolio rules in a continuous-time model. In *Stochastic optimization models in finance*, pages 621–661. Elsevier, 1971.

[164] Robert C Merton and Paul Anthony Samuelson. Continuous-time finance. 1992.

[165] M. Meyer, M. Desbrun, P. Schröder, and A. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*. Springer, 2003.

[166] 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. URL https://doi.org/10.1145/3592400.

[167] Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. Walkin' Robin: Walk on stars with Robin boundary conditions. *ACM Trans. Graph.*, 43(4), 2024. doi: 10.1145/3658153. URL https://doi.org/10.1145/3658153.

[168] GN Mil'shtejn. Approximate integration of stochastic differential equations. *Theory of Probability & Its Applications*, 19(3):557–562, 1975.

[169] Zackary Misso, Yining Karl Li, Brent Burley, Daniel Teece, and Wojciech Jarosz. Progressive null-tracking for volumetric rendering. In *ACM SIGGRAPH Conference Papers*, July 2023. doi: 10/kmdw.

[170] Jan Kloppenborg Møller and Henrik Madsen. *From state dependent diffusion to constant diffusion in stochastic differential equations by the Lamperti transform*. DTU Informatics, 2010.

[171] Jonathan T. Moon, Bruce Walter, and Stephen R. Marschner. Rendering discrete random media using precomputed scattering solutions. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, page 231–242, Goslar, DEU, 2007. Eurographics Association. ISBN 9783905673524.

[172] Jean-Paul Morillon. Numerical solutions of linear mixed boundary value problems using stochastic representations. *International journal for numerical methods in engineering*, 40(3): 387–405, 1997.

[173] Linus Mossberg. *GPU-Accelerated Monte Carlo Geometry Processing for Gradient-Domain Methods*. PhD thesis, Linköping University, Linköping, Sweden, 2021.

[174] Mervin E. Muller. Some Continuous Monte Carlo Methods for the Dirichlet Problem. *The Annals of Mathematical Statistics*, 27(3):569 – 589, 1956. doi: 10.1214/aoms/1177728169. URL https://doi.org/10.1214/aoms/1177728169.

[175] Thomas Müller, Markus Gross, and Jan Novák. Practical path guiding for efficient light-transport simulation. In *Comp. Graph. Forum*, volume 36. Wiley Online Library, 2017.

[176] Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Trans. Graph.*, 38(5), 2019.

[177] Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. Neural control variates. *ACM Trans. Graph.*, 39(6), 2020.

[178] Nathan Myhrvold and Francisco J Migoya. *Modernist bread*. Cooking Lab, 2017.

[179] Mohammad Sina Nabizadeh, Ravi Ramamoorthi, and Albert Chern. Kelvin transformations for simulations on infinite domains. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459809. URL https://doi.org/10.1145/3450626.3459809.

[180] NASA. Guidelines for thermal analysis of spacecraft hardware, 1999. URL https://llis.nasa.gov/lesson/695.

[181] A. Nealen. An as-short-as-possible intro to moving least squares. 2004. URL http://www.nealen.com/projects/mls/asapmls.pdf.

[182] Jean-Claude Nédélec. *Acoustic and electromagnetic equations: integral representations for harmonic problems*, volume 144. Springer, 2001.

[183] N. Newton. Variance reduction for simulated diffusions. *SIAM journal on applied mathematics*, 54(6):1780–1805, 1994.

[184] Fabrice Neyret. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Trans. Vis. Comp. Graph.*, 4(1), 1998.

[185] Vinh Phu Nguyen, Timon Rabczuk, Stéphane Bordas, and Marc Duflot. Meshless methods: a review and computer implementation aspects. *Mathematics and computers in simulation*, 79 (3):763–813, 2008.

[186] Baptiste Nicolet, Fabrice Rousselle, Jan Novak, Alexander Keller, Wenzel Jakob, and Thomas Müller. Recursive control variates for inverse rendering. *ACM Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301. doi: 10.1145/3592139. URL https://doi.org/10.1145/3592139.

[187] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: an adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph.*, 39(4), aug 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392406. URL https://doi.org/10.1145/3386569.3392406.

[188] 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.

[189] NVIDIA. Nvidia optix ai-accelerated denoiser. https://developer.nvidia.com/optix-denoiser, 2017.

[190] NVIDIA. NVIDIA real-time denoisers (NRD). https://developer.nvidia.com/rtx/ray-tracing/rt-denoisers, 2022.

[191] Bernt Øksendal. *Stochastic differential equations*. Springer Berlin, Heidelberg, 2003. ISBN 978-3-642-14394-6.

[192] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin. Diffusion curves: a vector representation for smooth-shaded images. In *ACM Transactions on Graphics*, volume 27, page 92. ACM, 2008.

[193] Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. ReSTIR GI: Path resampling for real-time path tracing. In *Computer Graphics Forum*, volume 40, pages 17–29. Wiley Online Library, 2021.

[194] Ozgur Ozdemir. Variable permittivity dielectric material loaded stepped-horn antenna. 2005.

[195] S. Papanicolopulos and A. Zervos. Polynomial c1 shape functions on the triangle. *Computers & Structures*, 118:53–58, 2013.

[196] Etienne Pardoux and Shanjian Tang. Forward-backward stochastic differential equations and quasilinear parabolic PDEs. *Probability Theory and Related Fields*, 114(2):123–150, 1999.

[197] S. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, et al. Optix: a general purpose ray tracing engine. In *ACM Transactions on Graphics*, volume 29, page 66. ACM, 2010.

[198] Paul William Partridge, Carlos Alberto Brebbia, et al. *Dual reciprocity boundary element method*. 2012.

[199] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J. Guibas. Meshless animation of fracturing solids. *ACM Trans. Graph.*, 24(3):957–964, jul 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073296. URL https://doi.org/10.1145/1073204.1073296.

[200] D. Pavić, M. Campen, and L. Kobbelt. Hybrid booleans. In *Computer Graphics Forum*, volume 29, pages 75–87. Wiley Online Library, 2010.

[201] J. Peters and U. Reif. *Subdivision Surfaces*, pages 57–81. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-76406-9. doi: 10.1007/978-3-540-76406-9_4.

[202] Susanne Pfalzner and Paul Gibbon. *Many-body tree methods in physics*. Cambridge University Press, 1997.

[203] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2016. ISBN 0128006455.

[204] Michel Pierre and Antoine Henrot. Shape variation and optimization. a geometrical analysis., 2018.

[205] Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, Yu-Kun Lai, and Shi-Min Hu. Principal curvatures from the integral invariant viewpoint. *Computer Aided Geometric Design*, 24 (8-9):428–442, 2007.

[206] Yang Qi, Dario Seyb, Benedikt Bitterli, and Wojciech Jarosz. A bidirectional formulation for walk on spheres. *Computer Graphics Forum*, 41(4), 2022. doi: 10.1111/cgf.14586. URL https://par.nsf.gov/biblio/10381119.

[207] Matthias Raab, Daniel Seibert, and Alexander Keller. Unbiased global illumination with participating media. In Alexander Keller, Stefan Heinrich, and Harald Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods*, pages 591–605. Springer, 2008. ISBN 978-3-540-74496-2.

[208] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[209] A. Requicha and H. Voelcker. Constructive solid geometry. 1977.

[210] P. Ribeiro, H. de Campos Velho, and H. Lopes. Helmholtz-Hodge decomposition and the analysis of 2d vector field ensembles. *Comput. Graph.*, 55(C), April 2016. ISSN 0097-8493.

[211] 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 Trans. Graph.*, 41(6), nov 2022. ISSN 0730-0301. doi: 10.1145/3550454.3555450. URL https://doi.org/10.1145/3550454.3555450.

[212] S. Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2): 109–144, 1982.

[213] F. Rousselle, M. Manzi, and M. Zwicker. Robust denoising using feature and color information. In *Computer Graphics Forum*, volume 32, pages 121–130. Wiley Online Library, 2013.

[214] Mael Rouxel-Labbé. *Anisotropic mesh generation*. PhD thesis, Université Côte d'Azur, 2016.

[215] Karl K Sabelfeld. Application of the von Mises–Fisher distribution to random walk on spheres method for solving high-dimensional diffusion–advection–reaction equations. *Statistics & Probability Letters*, 138:137–142, 2018.

[216] Karl K. Sabelfeld. Random walk on rectangles and parallelepipeds algorithm for solving transient anisotropic drift-diffusion-reaction problems. *Monte Carlo Methods and Applications*, 25(2):131–146, 2019. doi: doi:10.1515/mcma-2019-2039. URL https://doi.org/10.1515/mcma-2019-2039.

[217] Karl K. Sabelfeld and Nikolai A. Simonov. *Random Walks on Boundary for Solving PDEs*. De Gruyter, Berlin, Boston, 1994. ISBN 9783110942026. doi: doi:10.1515/9783110942026. URL https://doi.org/10.1515/9783110942026.

[218] B Sapoval, M Filoche, K Karamanos, and R Brizzi. Can one hear the shape of an electrode? i. numerical study of the active zone in Laplacian transfer. *The European Physical Journal B-Condensed Matter and Complex Systems*, 9:739–753, 1999.

[219] B Sapoval, JS Andrade Jr, and M Filoche. Catalytic effectiveness of irregular interfaces and rough pores: the "land surveyor approximation". *Chemical engineering science*, 56(17): 5011–5023, 2001.

[220] Rohan Sawhney. FCPW: Fastest closest points in the west, 2021.

[221] Rohan Sawhney and Keenan Crane. Monte Carlo geometry processing: a grid-free approach to PDE-based methods on volumetric domains. *ACM Trans. Graph.*, 39(4), aug 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392374. URL https://doi.org/10.1145/3386569.3392374.

[222] Rohan Sawhney and Bailey Miller. Zombie: A grid-free Monte Carlo solver for PDEs, 2023.

[223] Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. Grid-free Monte Carlo for PDEs with spatially varying coefficients. *ACM Trans. Graph.*, 41(4), jul 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530134. URL https://doi.org/10.1145/3528223.3530134.

[224] 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. URL https://doi.org/10.1145/3592398.

[225] B. Schäling. *The boost C++ libraries*. XML Press, 2014. ISBN 9781937434366 1937434362.

[226] Christoph Schied, Christoph Peters, and Carsten Dachsbacher. Gradient estimation for real-time adaptive temporal filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2):1–16, 2018.

[227] T. Schneider, Y. Hu, J. Dumas, X. Gao, D. Panozzo, and D. Zorin. Decoupling simulation accuracy from mesh quality. *ACM Transactions on Graphics*, 37(6):280, 2019.

[228] Thomas BA Senior. Impedance boundary conditions for imperfectly conducting surfaces. *Applied Scientific Research, Section B*, 8(1):418–436, 1960.

[229] Vadim Shapiro and Igor Tsukanov. Meshfree simulation of deforming domains. *Computer-Aided Design*, 31(7):459–471, 1999.

[230] Prateek Sharma and Gregory W Hammett. Preserving monotonicity in anisotropic diffusion. *Journal of Computational Physics*, 227(1):123–142, 2007.

[231] Nicholas Sharp and Alec Jacobson. Spelunking the deep: Guaranteed queries on general neural implicit surfaces via range analysis. *ACM Trans. Graph.*, 41(4), jul 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530155. URL https://doi.org/10.1145/3528223.3530155.

[232] Evan Shellshear and Robin Ytterlid. Fast distance queries for triangles, lines, and points using sse instructions. *Journal of Computer Graphics Techniques Vol*, 3(4), 2014.

[233] C. Shen, J. O'Brien, and J. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM Siggraph 2005 Courses*, page 204. ACM, 2005.

[234] Hang Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2), feb 2015. ISSN 0098-3500. doi: 10.1145/2629697. URL https://doi.org/10.1145/2629697.

[235] Nikolai A Simonov. Monte Carlo methods for solving elliptic equations with boundary conditions containing the normal derivative. In *Doklady Mathematics*, volume 74, pages 656–659. Citeseer, 2006.

[236] Nikolai A Simonov. Walk-on-spheres algorithm for solving third boundary value problem. *Applied Mathematics Letters*, 64:156–161, 2017.

[237] Jure Slak and Gregor Kosec. On generation of node distributions for meshless PDE discretizations. *SIAM Journal on Scientific Computing*, 41(5):A3202–A3229, 2019.

[238] J. Snyder and A. Barr. Ray tracing complex models containing surface tessellations. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 119–128, New York, NY, USA, 1987. ACM. ISBN 0-89791-227-6.

[239] Cyril Soler, Ronak Molazem, and Kartic Subr. A theoretical analysis of compactness of the light transport operator. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393379. doi: 10.1145/3528233.3530725. URL https://doi.org/10.1145/3528233.3530725.

[240] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[241] O. Stein, E. Grinspun, A. Jacobson, and M. Wardetzky. A mixed finite element method with piecewise linear elements for the biharmonic equation on surfaces, 2019.

[242] Ryusuke Sugimoto, Terry Chen, Yiti Jiang, Christopher Batty, and Toshiya Hachisuka. A practical walk-on-boundary method for boundary value problems. *ACM Trans. Graph.*, 42 (4), jul 2023. ISSN 0730-0301. doi: 10.1145/3592109. URL https://doi.org/10.1145/3592109.

[243] Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. Application of a particle-in-cell method to solid mechanics. *Comp. phys. comms.*, 87(1-2):236–252, 1995.

[244] T. Sun, P. Thamjaroenporn, and C. Zheng. Fast multipole representation of diffusion curves and points. *ACM Transactions on Graphics*, 33(4):53–1, 2014.

[245] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[246] László Szirmay-Kalos, Balázs Tóth, and Milán Magdics. Free path sampling in high resolution inhomogeneous participating media. *Computer Graphics Forum*, 30(1):85–97, 2011. ISSN 01677055.

[247] Y. Tong, S. Lombeyda, A. Hirani, and M. Desbrun. Discrete multiscale vector field decomposition. *ACM Transactions on Graphics*, 22(3):445–452, July 2003. ISSN 0730-0301.

[248] Heang K Tuy and Lee Tan Tuy. Direct 2-d display of 3-d objects. *IEEE Computer Graphics and Applications*, 4(10):29–34, 1984.

[249] T. Ullrich, V. Settgast, U. Krispel, C. Fünfzig, and D. Fellner. Distance calculation between a point and a subdivision surface. In *Vision, Modeling, and Visualization 2007. Proceedings*, pages 161–169. Max Planck Institut für Informatik, Saarbrücken, 2007.

[250] MV Umansky, MS Day, and TD Rognlien. On numerical solution of strongly anisotropic diffusion equation on misaligned grids. *Numerical Heat Transfer, Part B: Fundamentals*, 47(6): 533–554, 2005.

[251] E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.

[252] E. Veach and L. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428. ACM, 1995.

[253] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*, pages 145–167. Springer, 1995.

[254] P. Virtanen, R. Gommers, and Contributors. Scipy 1.0–fundamental algorithms for scientific computing in python. *arXiv e-prints*, art. arXiv:1907.10121, Jul 2019.

[255] Alan V Von Arx and Adon Delgado Jr. Convective heat transfer on mars. In *AIP Conference Proceedings*, volume 217, pages 734–739. American Institute of Physics, 1991.

[256] Jiří Vorba and Jaroslav Křivánek. Adjoint-driven Russian roulette and splitting in light transport simulation. *ACM Transactions on Graphics (TOG)*, 35(4):42:1–42:11, July 2016. ISSN 0730-0301. doi: 10/f89mcz.

[257] Carsten Wächter and Nikolaus Binder. A fast and robust method for avoiding self-intersection. *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*, pages 77–85, 2019.

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

[259] I. Wald, S. Woop, C. Benthin, G. Johnson, and M. Ernst. Embree: a kernel framework for efficient CPU ray tracing. *ACM Transactions on Graphics*, 33(4):143, 2014.

[260] I. Wald, W. Usher, N. Morrical, L. Lediaev, and V. Pascucci. RTX beyond ray tracing: Exploring the use of hardware ray tracing cores for tet-mesh point location. *Proceedings of High Performance Graphics*, 2019.

[261] Ingo Wald. On fast construction of sah-based bounding volume hierarchies. In *2007 IEEE Symposium on Interactive Ray Tracing*, pages 33–40. IEEE, 2007.

[262] Alastair J Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 8(10):127–128, 1974.

[263] Alastair J Walker. An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.*, 3(3):253–256, sep 1977. ISSN 0098-3500. doi: 10.1145/355744.355749. URL https://doi.org/10.1145/355744.355749.

[264] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P Greenberg. Lightcuts: A scalable approach to illumination. *ACM Trans. Graph.*, 24(3):1098–1107, jul 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073318. URL https://doi.org/10.1145/1073204.1073318.

[265] G. Ward and P. Heckbert. Irradiance gradients. Technical report, Lawrence Berkeley Lab., CA (United States); Ecole Polytechnique Federale . . . , 1992.

[266] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph.*, 22(4):85–92, jun 1988. ISSN 0097-8930. doi: 10.1145/378456.378490. URL https://doi.org/10.1145/378456.378490.

[267] BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[268] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23 (6):343–349, jun 1980. ISSN 0001-0782. doi: 10.1145/358876.358882. URL https://doi.org/10.1145/358876.358882.

[269] Mathias Willmann, Jesus Carrera, Xavier Sanchez-Vila, O Silva, and Marco Dentz. Coupling of mass transfer and reactive transport for nonlinear reactions in heterogeneous media. *Water resources research*, 46(7), 2010.

[270] E. R. Woodcock, T. Murphy, P. J. Hemmings, and T. C. Longworth. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Applications of Computing Methods to Reactor Problems*. Argonne National Laboratory, 1965.

[271] Lifan Wu, Shuang Zhao, Ling-Qi Yan, and Ravi Ramamoorthi. Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces. *ACM Trans. Graph.*, 38 (4), jul 2019. ISSN 0730-0301. doi: 10.1145/3306346.3322936. URL https://doi.org/10.1145/3306346.3322936.

[272] Ekrem Fatih Yılmazer, Delio Vicini, and Wenzel Jakob. Solving inverse PDE problems using grid-free Monte Carlo estimators. *arXiv preprint arXiv:2208.02114*, 2022.

[273] R. Ytterlid and E. Shellshear. BVH split strategies for fast distance queries. *Journal of Computer Graphics Techniques (JCGT)*, 4:1–25, 2015.

[274] Yonghao Yue, Kei Iwasaki, Bing-Yu Chen, Yoshinori Dobashi, and Tomoyuki Nishita. Toward optimal space partitioning for unbiased, adaptive free path sampling of inhomogeneous participating media. *Computer Graphics Forum*, 30(7):1911–1919, 2011.

[275] Cem Yuksel. Stochastic lightcuts. In *Proceedings of the Conference on High-Performance Graphics*, HPG '19, page 27–32, Goslar, DEU, 2019. Eurographics Association. doi: 10.2312/hpg.20191192. URL https://doi.org/10.2312/hpg.20191192.

[276] Laurent Zalewski, Stéphane Lassue, Daniel Rousse, and Kamel Boukhalfa. Experimental and numerical characterization of thermal bridges in prefabricated building walls. *Energy Conversion and Management*, 51(12):2869–2877, 2010.

[277] Yong Zhan, Sanjay V Kumar, Sachin S Sapatnekar, et al. Thermally aware design. *Foundations and Trends® in Electronic Design Automation*, 2(3):255–370, 2008.

[278] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4), aug 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392383. URL https://doi.org/10.1145/3386569.3392383.

[279] R. Zhao, M. Desbrun, G. Wei, and Y. Tong. 3d Hodge decompositions of edge- and face-based vector fields. *ACM Transactions on Graphics*, 38(5), 2019.

[280] Q. Zhou and A. Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.

[281] Q. Zhou, E. Grinspun, D. Zorin, and A. Jacobson. Mesh arrangements for solid geometry. *ACM Transactions on Graphics*, 35(4):1–15, 2016.

[282] Yijing Zhou and Wei Cai. Numerical solution of the Robin problem of Laplace equations with a Feynman–Kac formula and reflecting brownian motions. *Journal of Scientific Computing*, 69, 2016. URL https://doi.org/10.1007/s10915-016-0184-y.

[283] Yijing Zhou, Wei Cai, and Elton Hsu. Computation of the local time of reflecting Brownian motion and the probabilistic representation of the Neumann problem. *Communications in Mathematical Sciences*, 15(1):237–259, 2017. ISSN 1539-6746.

[284] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)*, 24(3):965–972, 2005.

[285] Olgierd Cecil Zienkiewicz and Jian Zhong Zhu. The superconvergent patch recovery and a posteriori error estimates. part 1: The recovery technique. *International Journal for Numerical Methods in Engineering*, 33(7):1331–1364, 1992.

[286] Olgierd Cecil Zienkiewicz and Jian Zhong Zhu. The superconvergent patch recovery and a posteriori error estimates. part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering*, 33(7):1365–1382, 1992.

[287] Chen Zong, Jiacheng Xu, Jiantao Song, Shuangmin Chen, Shiqing Xin, Wenping Wang, and Changhe Tu. P2M: A fast solver for querying distance from point to mesh surface. *ACM Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301. doi: 10.1145/3592439. URL https://doi.org/10.1145/3592439.

[288] M. Zwicker, W. Jarosz, J. Lehtinen, B. Moon, R. Ramamoorthi, F. Rousselle, P. Sen, C. Soler, and S. Yoon. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)*, 34 (2):667–681, May 2015.

Appendix A

Green's Functions & Their Derivatives

We provide expressions for the Green's function and its derivatives, in free space and over a ball in 2D and 3D, for the Poisson and screened Poisson equations. The PDE estimators we describe in Ch. 3 - 7 require these functions. We also discuss how to draw samples from the Green's function of a ball to reduce noise associated with the source term in these estimators.

A.1 Poisson Equation

A.1.1 Free Space Expressions

The free space Green's functions in 2D and 3D for two points x and y equal

$$G^{\mathbb{R}^2}(x,y) = -\frac{\log(r)}{2\pi}, \quad G^{\mathbb{R}^3}(x,y) = \frac{1}{4\pi r}, \tag{A.1}$$

where $r \coloneqq \|y - x\|$. The corresponding Poisson kernels equal

$$P^{\mathbb{R}^2}(x,y) = \frac{n_y \cdot (y-x)}{2\pi r^2}, \quad P^{\mathbb{R}^3}(x,y) = \frac{n_y \cdot (y-x)}{4\pi r^3}, \tag{A.2}$$

where n_y is the unit normal at y. The gradients of G with respect to x are

$$\frac{\partial G^{\mathbb{R}^2}(x,y)}{\partial x} = \frac{x-y}{2\pi r^2}, \qquad \frac{\partial G^{\mathbb{R}^3}(x,y)}{\partial x} = \frac{x-y}{4\pi r^3}. \tag{A.3}$$

Expressions for $\partial^2 G / \partial x \, \partial n_y$ are given by

$$\begin{aligned} \frac{\partial^2 G^{\mathbb{R}^2}(x,y)}{\partial x \, \partial n_y} &= 2\frac{n_y \cdot (y-x)}{2\pi r^4}(y-x) \; - \; \frac{n_y}{2\pi r^2}, \\ \frac{\partial^2 G^{\mathbb{R}^3}(x,y)}{\partial x \, \partial n_y} &= 3\frac{n_y \cdot (y-x)}{4\pi r^5}(y-x) \; - \; \frac{n_y}{4\pi r^3}. \end{aligned} \tag{A.4}$$

A.1.2 Expressions For A Ball

For a ball $\mathrm{B}(x,R)$ centered at x and with radius R, we can derive 2D and 3D Green's functions from the corresponding free space expressions using the *method of images* [55]. This gives

$$G^{\mathrm{B}}_{2\mathrm{D}}(x,y) = \frac{\log(R/r)}{2\pi}, \quad G^{\mathrm{B}}_{3\mathrm{D}}(x,y) = \frac{1}{4\pi}\left(\frac{1}{r} - \frac{1}{R}\right). \tag{A.5}$$

These functions integrate over B to

$$|G^{\mathrm{B}}_{2\mathrm{D}}(x)| \coloneqq \int_{\mathrm{B}(x,R)} G^{\mathrm{B}}_{2\mathrm{D}}(x,y)\,\mathrm{d}y = \frac{R^2}{4},$$

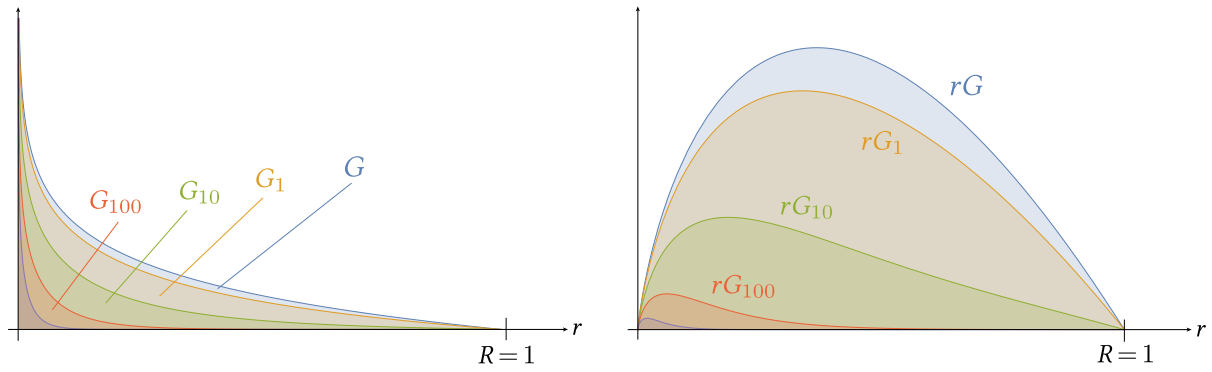


Figure A.1: To importance sample the Green's function of the Poisson and screened Poisson equations, we need to sample from densities that depend only on the radius r (shown here for 2D functions, and for several values of the screening parameter σ). Although Green's functions G have singularities at $r = 0$ (left), the associated radial distributions rG are nonsingular due to a change in measure (right).

$$|G^{\mathrm{B}}_{3\mathrm{D}}(x)| := \int_{\mathrm{B}(x,R)} G^{\mathrm{B}}_{3\mathrm{D}}(x,y)\,\mathrm{d}y = \frac{R^2}{6}. \tag{A.6}$$

The corresponding Poisson kernels are the same as their free space counterparts. For any point $z \in \partial\mathrm{B}$ with normal $n_z = (z-x)/R$, the Poisson kernels simplify to

$$P^{\mathrm{B}}_{2\mathrm{D}}(x,z) = \frac{1}{2\pi R}, \quad P^{\mathrm{B}}_{3\mathrm{D}}(x,z) = \frac{1}{4\pi R^2}. \tag{A.7}$$

Finally, gradients of G and P with respect to x are given by

$$\begin{aligned}
\frac{\partial G^{\mathrm{B}}_{2\mathrm{D}}(x,y)}{\partial x} &= \frac{(y-x)}{2\pi}\left(\frac{1}{r^2} - \frac{1}{R^2}\right), \\
\frac{\partial G^{\mathrm{B}}_{3\mathrm{D}}(x,y)}{\partial x} &= \frac{(y-x)}{4\pi}\left(\frac{1}{r^3} - \frac{1}{R^3}\right), \\
\frac{\partial P^{\mathrm{B}}_{2\mathrm{D}}(x,z)}{\partial x} &= 2\frac{(z-x)}{2\pi R^3}, \\
\frac{\partial P^{\mathrm{B}}_{3\mathrm{D}}(x,z)}{\partial x} &= 3\frac{(z-x)}{4\pi R^4}.
\end{aligned} \tag{A.8}$$

Sampling

As discussed in Sec. 4.2.2, we can use direction sampling to importance sample the Poisson kernel of a ball $P^B \equiv P^{\mathbb{R}^N}$. To generate samples from the probability density $p^{\mathrm{B}} := G^{\mathrm{B}}(x,y)/|G^{\mathrm{B}}(x)|$ associated with the Green's function of the ball $\mathrm{B}(x,R)$, we first pick a direction v uniformly on the unit sphere [5], and then use rejection sampling (Sec. 3.1.2) to sample a radius r from the density $2\pi r p^{\mathrm{B}}$ in 2D or $4\pi r^2 p^{\mathrm{B}}$ in 3D. The extra terms in front of p^{B} account for the change of measure between polar and Cartesian coordinates (Fig. A.1). We use $1.5/R$ as our rejection sampling bound for the radial density. The final sample point inside B is then given by $y = x + rv$. In 3D, we can instead use *Ulrich's polar method* [54, Sec. 9.4] to more efficiently sample the radius r via inversion (Sec. 3.1.2)—we refer to Mossberg [173, Sec. 2.5] for further details.

A.2 Screened Poisson Equation

A.2.1 Free Space Expressions

Let I_n and K_n (for $n = 0, 1, \ldots$) denote modified Bessel functions of the first and second kind, respectively—routines to efficiently evaluate these functions are available in numerical libraries such as *Boost* [225] and *SciPy* [254]. The free-space Green's functions in 2D and 3D for a screened Poisson equation with a positive screening coefficient σ then equal

$$G^{\sigma,\mathbb{R}^2}(x,y) = \frac{K_0(r\sqrt{\sigma})}{2\pi}, \quad G^{\sigma,\mathbb{R}^3}(x,y) = \frac{e^{-r\sqrt{\sigma}}}{4\pi r}. \tag{A.9}$$

The corresponding Poisson kernels are

$$\begin{aligned} P^{\sigma,\mathbb{R}^2}(x,y) &= Q^{\sigma,\mathbb{R}^2}(x,y)\ P^{\mathbb{R}^2}(x,y), \\ P^{\sigma,\mathbb{R}^3}(x,y) &= Q^{\sigma,\mathbb{R}^3}(x,y)\ P^{\mathbb{R}^3}(x,y), \end{aligned} \tag{A.10}$$

where

$$\begin{aligned} Q^{\sigma,\mathbb{R}^2}(x,y) &\coloneqq K_1(r\sqrt{\sigma})\ r\sqrt{\sigma}, \\ Q^{\sigma,\mathbb{R}^3}(x,y) &\coloneqq e^{-r\sqrt{\sigma}}\left(r\sqrt{\sigma}+1\right). \end{aligned} \tag{A.11}$$

The gradients of G^σ with respect to x are

$$\begin{aligned} \frac{\partial G^{\sigma,\mathbb{R}^2}(x,y)}{\partial x} &= Q^{\sigma,\mathbb{R}^2}(x,y)\ \frac{\partial G^{\mathbb{R}^2}(x,y)}{\partial x}, \\ \frac{\partial G^{\sigma,\mathbb{R}^3}(x,y)}{\partial x} &= Q^{\sigma,\mathbb{R}^3}(x,y)\ \frac{\partial G^{\mathbb{R}^3}(x,y)}{\partial x}. \end{aligned} \tag{A.12}$$

Applying the product rule to the expressions above yields $\partial^2 G^\sigma / \partial x\ \partial n_y$.

A.2.2 Expressions For A Ball

Centered Expressions

For a ball $\mathrm{B}(x,R)$, the 2D and 3D Green's functions are

$$\begin{aligned} G^{\sigma,\mathrm{B}}_{2\mathrm{D}}(x,y) &= \frac{1}{2\pi} V^{\sigma,\mathrm{B}}_{2\mathrm{D}}(\|y-x\|), \quad \text{where } V^{\sigma,\mathrm{B}}_{2\mathrm{D}}(r) \coloneqq \left(K_0(r\sqrt{\sigma}) - I_0(r\sqrt{\sigma})\frac{K_0(R\sqrt{\sigma})}{I_0(R\sqrt{\sigma})}\right), \\ G^{\sigma,\mathrm{B}}_{3\mathrm{D}}(x,y) &= \frac{1}{4\pi}\sqrt{\frac{2\sqrt{\sigma}}{\pi r}}\left(K_{\frac{1}{2}}(r\sqrt{\sigma}) - \frac{K_{\frac{1}{2}}(R\sqrt{\sigma})}{I_{\frac{1}{2}}(R\sqrt{\sigma})} I_{\frac{1}{2}}(r\sqrt{\sigma})\right) \\ &= \frac{1}{4\pi}\left(\frac{e^{-r\sqrt{\sigma}}}{r} - \frac{e^{-R\sqrt{\sigma}}}{R}\left(\frac{\sinh(r\sqrt{\sigma})}{r\sqrt{\sigma}}\frac{R\sqrt{\sigma}}{\sinh(R\sqrt{\sigma})}\right)\right) \\ &= \frac{1}{4\pi} V^{\sigma,\mathrm{B}}_{3\mathrm{D}}(r), \quad \text{where } V^{\sigma,\mathrm{B}}_{3\mathrm{D}}(r) \coloneqq \left(\frac{\sinh((R-r)\sqrt{\sigma})}{r\sinh(R\sqrt{\sigma})}\right). \end{aligned} \tag{A.13}$$

They integrate over B to

$$|G^{\sigma,\mathrm{B}}_{2\mathrm{D}}(x)| \coloneqq \int_{\mathrm{B}(x,R)} G^{\sigma,\mathrm{B}}_{2\mathrm{D}}(x,y)\,\mathrm{d}y = \frac{1}{\sigma}\left(1 - \frac{1}{I_0(R\sqrt{\sigma})}\right),$$

$$|G_{3\mathrm{D}}^{\sigma,\mathrm{B}}(x)| \coloneqq \int_{\mathrm{B}(x,R)} G_{3\mathrm{D}}^{\sigma,\mathrm{B}}(x,y)\,\mathrm{d}y = \frac{1}{\sigma}\left(1 - \frac{R\sqrt{\sigma}}{\sinh(R\sqrt{\sigma})}\right). \tag{A.14}$$

The corresponding Poisson kernels at any point $y \in \mathrm{B}$ equal

$$\begin{aligned} P_{2\mathrm{D}}^{\sigma,\mathrm{B}}(x,y) &= Q_{2\mathrm{D}}^{\sigma,\mathrm{B}}(x,y)\ P^{\mathbb{R}^2}(x,y), \\ P_{3\mathrm{D}}^{\sigma,\mathrm{B}}(x,y) &= Q_{3\mathrm{D}}^{\sigma,\mathrm{B}}(x,y)\ P^{\mathbb{R}^3}(x,y), \end{aligned} \tag{A.15}$$

where

$$\begin{aligned} Q_{2\mathrm{D}}^{\sigma,\mathrm{B}}(x,y) &\coloneqq \left(K_1(r\sqrt{\sigma}) + I_1(r\sqrt{\sigma})\frac{K_0(R\sqrt{\sigma})}{I_0(R\sqrt{\sigma})}\right)\ r\sqrt{\sigma}, \\ Q_{3\mathrm{D}}^{\sigma,\mathrm{B}}(x,y) &\coloneqq e^{-r\sqrt{\sigma}}\left(r\sqrt{\sigma}+1\right)\ + \\ &\qquad \left(\cosh(r\sqrt{\sigma})\ r\sqrt{\sigma} - \sinh(r\sqrt{\sigma})\right)\frac{e^{-R\sqrt{\sigma}}}{\sinh(R\sqrt{\sigma})}. \end{aligned} \tag{A.16}$$

We note that $Q^{\sigma,\mathrm{B}} \in [0,1)$ for $\sigma > 0$. Since WoSt samples directions proportionally to $P^{\mathbb{R}^N}$ (Sec. 4.2.2), we are left with a multiplicative weight of $Q^{\sigma,\mathrm{B}}$ in the solution estimate for a screened Poisson equation at every step of a random walk—as mentioned in Sec. 3.2.2, we can apply Russian roulette on $1 - Q^{\sigma,\mathrm{B}}$ to terminate walks early.

For any point $z \in \partial\mathrm{B}$ with normal n_z, the Poisson kernels simplify to

$$\begin{aligned} P_{2\mathrm{D}}^{\sigma,\mathrm{B}}(x,z) &= \frac{1}{2\pi R}\left(\frac{1}{I_0(R\sqrt{\sigma})}\right), \\ P_{3\mathrm{D}}^{\sigma,\mathrm{B}}(x,z) &= \frac{1}{4\pi R^2}\left(\frac{R\sqrt{\sigma}}{\sinh(R\sqrt{\sigma})}\right). \end{aligned} \tag{A.17}$$

Notice that in both dimensions, the Poisson kernel equals $\frac{1-\sigma|G^{\sigma,\mathrm{B}}(x)|}{|\partial\mathrm{B}(x,R)|}$, and

$$\sigma|G^{\sigma,\mathrm{B}}(x)| + |P^{\sigma,\mathrm{B}}(x)| = 1, \tag{A.18}$$

where $|P^{\sigma,\mathrm{B}}(x)|$ denotes the integral of the Poisson kernel over B. Intuitively, a random walk is either absorbed inside the ball *or* it escapes through the boundary. We use these facts to develop the delta tracking variant of WoS in Sec. 5.2.1.

Finally, gradients of G^σ and P^σ with respect to x are given by

$$\begin{aligned} \frac{\partial G_{2\mathrm{D}}^{\sigma,\mathrm{B}}(x,y)}{\partial x} &= \frac{(y-x)\sqrt{\sigma}}{2\pi r}\left(K_1(r\sqrt{\sigma}) - \frac{K_1(R\sqrt{\sigma})}{I_1(R\sqrt{\sigma})}I_1(r\sqrt{\sigma})\right), \\ \frac{\partial G_{3\mathrm{D}}^{\sigma,\mathrm{B}}(x,y)}{\partial x} &= \frac{(y-x)\sqrt{\sigma}}{4\pi r^2}\left(e^{-r\sqrt{\sigma}}\left(1+\frac{1}{r\sqrt{\sigma}}\right) - \right. \\ &\quad \left.\left(\cosh(r\sqrt{\sigma}) - \frac{\sinh(r\sqrt{\sigma})}{r\sqrt{\sigma}}\right)\left(\frac{e^{-R\sqrt{\sigma}}\left(1+\frac{1}{R\sqrt{\sigma}}\right)}{\cosh(R\sqrt{\sigma}) - \frac{\sinh(R\sqrt{\sigma})}{R\sqrt{\sigma}}}\right)\right), \\ \frac{\partial P_{2\mathrm{D}}^{\sigma,\mathrm{B}}(x,z)}{\partial x} &= \frac{(z-x)\ \sigma}{2\pi R}\left(\frac{1}{R\sqrt{\sigma}\ I_1(R\sqrt{\sigma})}\right), \\ \frac{\partial P_{3\mathrm{D}}^{\sigma,\mathrm{B}}(x,z)}{\partial x} &= \frac{(z-x)\ \sigma}{4\pi R^2}\left(\frac{1}{\cosh(R\sqrt{\sigma}) - \frac{\sinh(R\sqrt{\sigma})}{R\sqrt{\sigma}}}\right). \end{aligned} \tag{A.19}$$

Off-Centered Expressions

The next-flight variant of WoS in Sec. 5.2.2 requires off-centered versions of the Green's function and Poisson kernel. In particular, let x be an arbitrary point inside a ball $\mathrm{B}(c, R)$ centered at c, and let $r_- \coloneqq \min(|x-c|, |y-c|)$ and $r_+ \coloneqq \max(|x-c|, |y-c|)$. Furthermore, let θ define the angle between the vectors $x-c$ and $y-c$ in 2D or 3D. Then the off-centered Green's function is given by the infinite series

$$\begin{aligned} G^{\sigma,\mathrm{B}}_{2\mathrm{D}}(x,y) &= \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} \cos(n\theta)\ I_n(r_-\sqrt{\sigma}) \left(K_n(r_+\sqrt{\sigma}) - \frac{K_n(R\sqrt{\sigma})}{I_n(R\sqrt{\sigma})} I_n(r_+\sqrt{\sigma}) \right), \\ G^{\sigma,\mathrm{B}}_{3\mathrm{D}}(x,y) &= \frac{1}{4\pi} \sum_{n=0}^{\infty} (2n+1)\ P_n(\cos(\theta)) \left(\sqrt{\frac{\pi}{2r_-\sqrt{\sigma}}} I_{n+\frac{1}{2}}(r_-\sqrt{\sigma}) \right) \\ &\qquad \sqrt{\frac{2\sqrt{\sigma}}{\pi r_+}} \left(K_{n+\frac{1}{2}}(r_+\sqrt{\sigma}) - \frac{K_{n+\frac{1}{2}}(R\sqrt{\sigma})}{I_{n+\frac{1}{2}}(R\sqrt{\sigma})} I_{n+\frac{1}{2}}(r_+\sqrt{\sigma}) \right), \end{aligned} \tag{A.20}$$

where P_n denotes the recursively defined Legendre polynomials. The Poisson kernel can be computed by evaluating $n_z \cdot \partial G^\sigma(x,z)/\partial z$ on $\partial\mathrm{B}$. We recover the centered expressions for G^σ and P^σ from the previous section when x coincides with c.

In practice, we observe that 100 to 200 terms are required to accurately approximate these series. To avoid this computational burden, we provide approximations for off-centered functions. In particular, let $u \coloneqq x-c$, $v \coloneqq y-c$ and $w \coloneqq y-x$. Then in 2D and 3D we have

$$\begin{aligned} G^{\sigma,\mathrm{B}}_{2\mathrm{D}}(x,y) &= \frac{1}{2\pi} \left(V^\sigma_{2\mathrm{D}}(|w|) - V^\sigma_{2\mathrm{D}} \left(\frac{R^2 - u\cdot v}{R} \right) \right), \\ G^{\sigma,\mathrm{B}}_{3\mathrm{D}}(x,y) &= \frac{1}{4\pi} \left(V^\sigma_{3\mathrm{D}}(|w|) - V^\sigma_{3\mathrm{D}} \left(\frac{R^2 - u\cdot v}{R} \right) \right), \\ P^{\sigma,\mathrm{B}}_{2\mathrm{D}}(x,y) &= \frac{1}{2\pi} \left(W^{\sigma,\mathrm{B}}_{2\mathrm{D}}(|w|) \frac{|v|^2 - u\cdot v}{|w||v|} + W^{\sigma,\mathrm{B}}_{2\mathrm{D}} \left(\frac{R^2 - u\cdot v}{R} \right) \frac{u\cdot v}{R|v|} \right), \\ P^{\sigma,\mathrm{B}}_{3\mathrm{D}}(x,y) &= \frac{1}{4\pi} \left(W^{\sigma,\mathrm{B}}_{3\mathrm{D}}(|w|) \frac{|v|^2 - u\cdot v}{|w||v|} + W^{\sigma,\mathrm{B}}_{3\mathrm{D}} \left(\frac{R^2 - u\cdot v}{R} \right) \frac{u\cdot v}{R|v|} \right), \end{aligned} \tag{A.21}$$

where

$$\begin{aligned} W^{\sigma,\mathrm{B}}_{2\mathrm{D}}(r) &\coloneqq \sqrt{\sigma} \left(K_1(r\sqrt{\sigma}) + \frac{K_0(R\sqrt{\sigma})}{I_0(R\sqrt{\sigma})} I_1(r\sqrt{\sigma}) \right), \\ W^{\sigma,\mathrm{B}}_{3\mathrm{D}}(r) &\coloneqq \sqrt{\sigma} \sqrt{\frac{2\sqrt{\sigma}}{\pi r}} \left(K_{\frac{3}{2}}(r\sqrt{\sigma}) + \frac{K_{\frac{1}{2}}(R\sqrt{\sigma})}{I_{\frac{1}{2}}(R\sqrt{\sigma})} I_{\frac{3}{2}}(r\sqrt{\sigma}) \right) \\ &= \frac{\sqrt{\sigma}}{r} \left(e^{-r\sqrt{\sigma}} \left(1 + \frac{1}{r\sqrt{\sigma}} \right) + \right. \\ &\qquad \left. \frac{e^{-R\sqrt{\sigma}}}{\sinh(R\sqrt{\sigma})} \left(\cosh(r\sqrt{\sigma}) - \frac{\sinh(r\sqrt{\sigma})}{r\sqrt{\sigma}} \right) \right). \end{aligned} \tag{A.22}$$

These expressions for G^σ and P^σ are exact when x coincides with c, but begin to diverge slightly from the true values as x is moved closer to $\partial\mathrm{B}$ and the value of the coefficient σ is decreased, as shown in Fig. A.2. In our experiments, we observed that these approximate expressions provide sufficiently accurate results with the next-flight variant of WoS with far less compute, especially when σ is large.

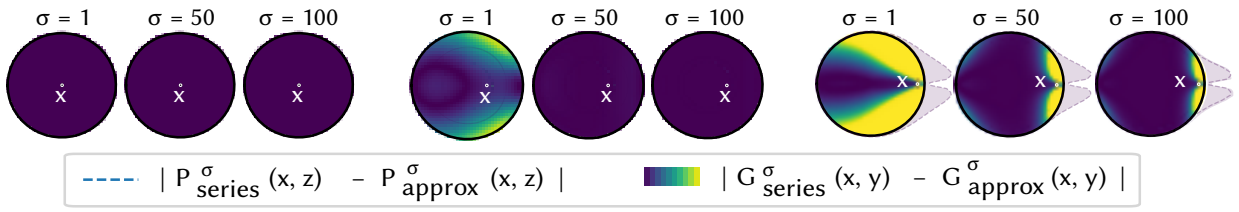


Figure A.2: The series and approximate expressions for $G^{\sigma,\mathrm{B}}$ and $P^{\sigma,\mathrm{B}}$ match exactly when x lies at the center of the ball B (left), but begin to diverge as x is moved closer to $\partial\mathrm{B}$ (middle) and σ is decreased (right).

Sampling

We follow App. A.1.2 to generate samples from the probability density $p^{\sigma,\mathrm{B}} := G^{\sigma,\mathrm{B}}(x,y)/|G^{\sigma,\mathrm{B}}(x)|$, but bound the radial density by the following case dependent function instead:

$$h(R,\sigma) := \begin{cases} \max(2.2 * \max(1/R, 1/\sigma), 0.6 * \max(\sqrt{R}, \sqrt{\sigma})) & R \leq \sigma, \\ \max(2.2 * \min(1/R, 1/\sigma), 0.6 * \min(\sqrt{R}, \sqrt{\sigma})) & \text{otherwise.} \end{cases} \tag{A.23}$$

Generating samples from the *off-centered* Green's functions in App. A.2.2 is more challenging as we do not have closed-form expressions for $|G^{\sigma,\mathrm{B}}(x)|$. Though we could use a uniform density $p^{\sigma,\mathrm{B}} := 1/|\mathrm{B}(c,R)|$ for unbiased estimation, we instead use weighted importance resampling (Sec. 3.1.2) to generate samples that approximately importance sample $G^{\sigma,\mathrm{B}}$.

Appendix B

Open Domains & Double-Sided Boundaries

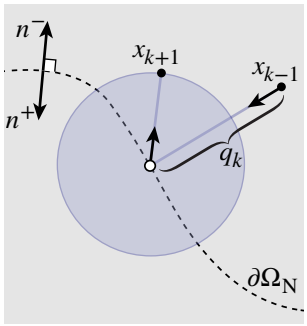
Here we describe how to apply walk on stars (Ch. 4) and boundary value caching (Sec. 7.5) to open domains, by adapting the boundary integral from Eq. 2.16 to use an appropriate set of Dirichlet, Neumann and Robin conditions on either side of the domain boundary.

B.1 Walk On Stars

Let $\Omega \in \mathbb{R}^N$ be an open domain, and let n^+ and n^- denote unit "outward" and "inward" facing normals on $\partial\Omega$ (respectively). We assume $\partial\Omega$ has a canonical orientation determined by n^+, and partition it into a Dirichlet part $\partial\Omega_{\mathrm{D}}$ with prescribed values g^+ and g^- on either side $\partial\Omega_{\mathrm{D}}$, and a Neumann part $\partial\Omega_{\mathrm{N}}$ with corresponding derivative values h^+ and h^- (which we will later generalize to Robin conditions). We then restrict Eq. 2.16 to a star-shaped region $\mathrm{St}(x, R)$ to get

$$\begin{aligned} \alpha(x)\, u(x) = &\int_{\partial \mathrm{St}_{\mathrm{B}}(x,R)} P^B(x,z)\, u(z)\, \mathrm{d}z \\ &+ \int_{\partial \mathrm{St}_{\mathrm{N}}(x,R)} P^{B,*}(x,z)\, u(z) \;+\; G^B(x,z)\, h^*(z)\, \mathrm{d}z \\ &+ \int_{\mathrm{St}(x,R)} G^B(x,y)\, f(y)\, \mathrm{d}y, \end{aligned} \tag{B.1}$$

where the Neumann data h^* equals either h^+ or h^- depending on where x is, and correspondingly $P^{B,*}(x,z) \coloneqq {}^{-\partial G^B(x,z)}/_{\partial n_z^*}$. Contributions from the solution $u(z)$ and derivative $h^*(z)$ on non-visible parts of $\partial\mathrm{B}$ and $\partial\Omega_{\mathrm{N}}$ for the region $\mathrm{B}(x,r) \cap \Omega$ can be ignored, as they define a closed region that does not contain x and hence integrate to zero (see Eq. 2.11). Inside a star-shaped region St, by construction we have $\alpha = 1/2$ if $x \in \partial\mathrm{St}_{\mathrm{N}}$, and 1 otherwise.



The choice of which boundary conditions to use during a random walk depends on whether the boundary is front- or back-facing relative to the walk locations x_k or x_{k-1} for $k > 0$, as well as the walk's direction of approach towards $\partial\Omega$, $q_k \coloneqq x_k - x_{k-1}$ (see inset). For $k = 0$, we require q_0 as input to the algorithm from the user to determine the appropriate boundary conditions—for instance when $x_0 \in \partial\Omega_{\mathrm{D}}$, we use the Dirichlet data g^+ if $q_0 = n^+_{x_0}$, and g^- if $q_0 = n^-_{x_0}$. More generally:

1. we change *line* 2 in Alg. 2 to return g^+ when $q_k \cdot n^+_{x_k} > 0$, and g^- otherwise. This dot product determines whether the boundary is back-facing relative to x_{k-1}.

2. before using hemispherical direction sampling in *line* 5 to determine the next walk location x_{k+1}, we flip the direction of the boundary normal n_x in Alg. 2 from $n^+_{x_k}$ to $n^-_{x_k}$ if $q_k \cdot n^+_{x_k} < 0$. This ensures that x_{k+1} will lie on the same side of the boundary as the direction from which the walk approached $\partial\Omega_{\mathrm{N}}$.

3. for Neumann data:

    a) when $x_k \notin \partial\Omega_{\mathrm{N}}$, we use h^+ at the sampled point z_{k+1} if $(z_{k+1} - x_k) \cdot n^+_{z_{k+1}} > 0$, and h^- otherwise. The boundary orientation in this case is determined relative to x_k.

    b) when $x_k \in \partial\Omega_{\mathrm{N}}$, we use h^+ as the Neumann data at z_{k+1} if the boundary normal n_x was flipped as described in change 2, and h^- otherwise.

For Robin conditions with coefficients $\mu^+, \mu^- \in \mathbb{R}^N_{>0}$ on either side of $\partial\Omega_{\mathrm{R}}$, the only change to Eq. B.1 is to introduce a reflectance term $\rho^* := 1 - \mu^* G^B / P^{B,*}$ in front of $P^{B,*}(x, z)\, u(z)$. Here we use μ^+ inside the reflectance if $q_k \cdot n^+_{x_k} < 0$, and μ^- otherwise.

B.2 Boundary Value Caching

For sample caching and reuse, we work directly with the boundary integral in Eq. 2.16. Similar to Sec. 7.5, we generate uniformly distributed source samples over Ω, but now use two separate sets of boundary samples associated with the normals n^+ and n^- on $\partial\Omega$. We use WoSt to estimate solution values u^+ and u^- by launching random walks on either side of a Neumann boundary $\partial\Omega_{\mathrm{N}}$ and Robin boundary $\partial\Omega_{\mathrm{R}}$. We also generate boundary samples on two offset Dirichlet boundaries $\partial\Omega^{\zeta}_{\mathrm{D}}$ and $\partial\Omega^{-\zeta}_{\mathrm{D}}$ to estimate $\partial u^+/\partial n^+$ and $\partial u^-/\partial n^-$, respectively. With this setup, we can compute a solution estimate at any evaluation point $x \in \Omega$ by directly estimating Eq. 2.16 with all the source and boundary samples—for points that are within a distance ζ to $\partial\Omega_{\mathrm{D}}$, we instead use WoSt to compute pointwise solution estimates.

Appendix C

Operator-Theoretic Analysis

In this appendix, we provide a formal justification for the convergence of the walk on stars method (Ch. 4) with Dirichlet, Neumann and Robin conditions via an operator-theoretic analysis of boundary integral equations. We draw inspiration from the seminal work of Arvo [6], who pioneered the use of operator-theoretic techniques to analyze the convergence of integral equations in light transport—Arvo [7] and Veach [251, Ch. 4 & 7] provide a detailed treatment of this topic, and Soler et al. [239] describe recent developments.

C.1 Background

The material in this section is based on results by Jörgens [120, Ch. 2] and Kato [128, Ch. 3]. To simplify the discussion, we work with a Banach space $\mathscr{L}^2(\Omega)$ of square-integrable functions on a domain Ω, equipped with the usual functional norm $\|\cdot\|_{\mathscr{L}^2(\Omega)}$. *I.e.*, for any function $u \in \mathscr{L}^2(\Omega)$,

$$\|u\|_{\mathscr{L}^2(\Omega)} \coloneqq \left(\int_\Omega |u(x)|^2 \,\mathrm{d}x\right)^{\frac{1}{2}} \quad \text{and} \quad \|u\|_{\mathscr{L}^2(\Omega)} < \infty. \tag{C.1}$$

Even though the solution of linear elliptic PDEs like the Poisson equation generally exits in more restrictive spaces (*e.g.*, spaces requiring differentiability), working with such spaces requires more complicated techniques to arrive at the same results (*e.g.*, using different inequalities to bound operator norms).

Given a kernel function $\kappa : \Omega \times \Omega \to \mathbb{R}$ and a boundary mapping $\mathrm{A} : \Omega \to \mathscr{P}(\Omega)$ (where $\mathscr{P}$ denotes the powerset), we define an integral operator

$$\mathscr{C}_\kappa^{\mathrm{A}}[u](x) \coloneqq \int_{\mathrm{A}(x)} \kappa(x,y) u(y) \,\mathrm{d}y. \tag{C.2}$$

If the kernel and boundary mapping satisfy

$$\sup_{x\in\Omega} \int_{\mathrm{A}(x)} |\kappa(x,y)| \,\mathrm{d}y < C < \infty \tag{C.3}$$

for some constant C, then the integral operator $\mathscr{C}_\kappa^{\mathrm{A}}$ satisfies the following properties:

1. Its codomain is $\mathscr{L}^2(\Omega)$, *i.e.*, $\mathscr{C}_\kappa^{\mathrm{A}} : \mathscr{L}^2(\Omega) \to \mathscr{L}^2(\Omega)$.

2. It is bounded and in particular, for all $u \in \mathscr{L}^2(\Omega)$,

$$\left\|\mathscr{C}_\kappa^{\mathrm{A}}[u]\right\|_{\mathscr{L}^2(\Omega)} \leq C \left\|u\right\|_{\mathscr{L}^2(\Omega)}. \tag{C.4}$$

These two properties together imply that the operator has a well defined operator norm

$$\left\|\mathscr{C}_\kappa^{\mathrm{A}}\right\|_{\mathrm{Op}} := \sup\left\{\frac{\left\|\mathscr{C}_\kappa^{\mathrm{A}}[u]\right\|_{\mathscr{L}^2(\Omega)}}{\|u\|_{\mathscr{L}^2(\Omega)}}, u \in \mathscr{L}^2(\Omega), u \neq 0\right\}, \tag{C.5}$$

and spectral radius

$$r\left(\mathscr{C}_\kappa^{\mathrm{A}}\right) := \lim_{n\to\infty}\left\|\mathscr{C}_\kappa^{\mathrm{A}}\right\|_{\mathrm{Op}}^{\frac{1}{n}}. \tag{C.6}$$

It also follows that

$$\left\|\mathscr{C}_\kappa^{\mathrm{A}}\right\|_{\mathrm{Op}} \leq C \text{ and } r\left(\mathscr{C}_\kappa^{\mathrm{A}}\right) < C. \tag{C.7}$$

Moreover, for an operator $\mathscr{C}_\kappa^{\mathrm{A}}$, its unit resolvent is the operator

$$\mathscr{R}_\kappa^{\mathrm{A}} := \left(\mathscr{I} - \mathscr{C}_\kappa^{\mathrm{A}}\right)^{-1}, \tag{C.8}$$

where $\mathscr{I} : \mathscr{L}^2(\Omega) \to \mathscr{L}^2(\Omega)$ is the identity operator. The unit resolvent $\mathscr{R}_\kappa^{\mathrm{A}}$ exists and is a bounded operator $\mathscr{L}^2(\Omega) \to \mathscr{L}^2(\Omega)$ if and only if $r\left(\mathscr{C}_\kappa^{\mathrm{A}}\right) < 1$ (the inequality is strict).

C.2 Walk On Stars With Dirichlet-Neumann Conditions

To understand the convergence of WoSt with mixed Dirichlet-Neumann conditions, we will express the boundary integral in Eq. 4.1 in operator-theoretic form. We achieve this by defining the convolutional kernel $\kappa := P^{\mathrm{B}}$, the mapping $\mathrm{A} : x \mapsto \partial\mathrm{St}(x, R)$, and the corresponding integral operator $\mathscr{C}_P^{\mathrm{St}}$ (we leave the radius R unspecified for now, which means that the region $\mathrm{St}(x, R)$ is not necessarily star-shaped). This allows us to rewrite Eq. 4.1 equivalently in operator form as

$$u = \mathscr{C}_P^{\mathrm{St}}[u] + s, \tag{C.9}$$

where we use $s \in \mathscr{L}^2(\Omega)$ to denote the non-recursive terms on the right-hand side of Eq. 4.1. Eq. C.9 is a Fredholm-Volterra equation of the second kind, whose solution exists only if the unit resolvent $\mathscr{R}_P^{\mathrm{St}} := \left(\mathscr{I} - \mathscr{C}_P^{\mathrm{St}}\right)^{-1}$ of $\mathscr{C}_P^{\mathrm{St}}$ exists and is bounded [13, 65]. If $\mathscr{R}_P^{\mathrm{St}}$ is indeed bounded, then by rearranging terms we can write the solution as

$$u = \mathscr{R}_P^{\mathrm{St}}[s]. \tag{C.10}$$

To estimate Eq. C.9 using recursive Monte Carlo, we also require boundedness of $\mathscr{R}_P^{\mathrm{St}}$ for any estimator to be convergent and have finite variance [98]. Therefore, to ensure the WoSt estimator is practical, we need to select a radius R for region $\mathrm{St}(x, R)$ that guarantees this condition on $\mathscr{R}_P^{\mathrm{St}}$.

To this end, we take advantage of the fact that the Poisson kernel P^{B} is the *signed solid angle* kernel (Sec. 4.2.2) that integrates to 1 over any closed region [12, 114]. We therefore have

$$\int_{\partial\mathrm{St}(x,R)} |P^{\mathrm{B}}(x, y)| \, \mathrm{d}y \geq \int_{\partial\mathrm{St}(x,R)} P^{\mathrm{B}}(x, y) \, \mathrm{d}y = 1. \tag{C.11}$$

The inequality follows from basic properties of the absolute value and integration, and becomes an equality in regions $\mathrm{St}(x, R)$ where the Poisson kernel is positive for all $y \in \partial\mathrm{St}(x, R)$, *i.e.*, *star-shaped regions* where all points y are visible from x. By selecting R to be the minimum of the

distance to the closest silhouette point on the Neumann boundary and the closest point on the Dirichlet boundary, WoSt ensures that $\mathrm{St}(x, R)$ is star-shaped, and thus

$$\int_{\partial \mathrm{St}(x,R)} |P^{\mathrm{B}}(x, y)| \, \mathrm{d}y = 1. \tag{C.12}$$

Together with Eq. C.7, Eq. C.12 guarantees that the spectral radius $r\left(\mathscr{C}_P^{\mathrm{St}}\right) \leq 1$. However, for the unit resolvent to exist and be bounded, we require this inequality to be strict. This is achieved through the ε-shell approximation, which effectively replaces the Poisson kernel with a kernel $P_\varepsilon^{\mathrm{B}}$ such that $P_\varepsilon^{\mathrm{B}}(x, y) = 0$ for $y \in \partial \mathrm{St}(x, R) \cup \partial \Omega_{\mathrm{D}}^{\varepsilon}$, and $P_\varepsilon^{\mathrm{B}}(x, y) = P^{\mathrm{B}}(x, y)$ otherwise. Then,

$$\int_{\partial \mathrm{St}(x,R)} |P_\varepsilon^{\mathrm{B}}(x, y)| \, \mathrm{d}y < \int_{\partial \mathrm{St}(x,R)} |P^{\mathrm{B}}(x, y)| \, \mathrm{d}y = 1, \tag{C.13}$$

ensuring the unit resolvent exists. Since WoSt reduces to WoS for pure Dirichlet problems, the analysis in this section also applies to the latter as its spherical domains always satisfy Eq. C.12—in particular, this analysis explains the need for the ε-shell approximation in WoS to achieve convergence. In contrast, the WoSt estimator for pure Neumann problems does not converge to a unique solution without Tikhonov regularization, as a lack of a termination criterion means that the spectral radius $r\left(\mathscr{C}_P^{\mathrm{St}}\right)$ is not strictly less than 1.

C.3 Walk On Stars With Robin Conditions

Lastly, we consider WoSt with Robin conditions (Sec. 4.3). As above, we start by writing the boundary integral in Eq. 4.6 in operator-theoretic form, but we now modify our kernel to include the reflectance term, $\kappa := \rho_\mu P^{\mathrm{B}}$, and use the same mapping A. Using the corresponding integral operator $\mathscr{C}_{\rho_\mu P}^{\mathrm{St}}$, we rewrite Eq. 4.6 equivalently in operator form as

$$u = \mathscr{C}_{\rho_\mu P}^{\mathrm{St}}[u] + s, \tag{C.14}$$

As in the previous section, we need to select a radius R that ensures the unit resolvent $\mathscr{R}_{\rho_\mu P}^{\mathrm{St}} := \left(\mathscr{I} - \mathscr{C}_{\rho_\mu P}^{\mathrm{St}}\right)^{-1}$ of $\mathscr{C}_{\rho_\mu P}^{\mathrm{St}}$ exists and is bounded. This will guarantee that the solution

$$u = \mathscr{R}_{\rho_\mu P}^{\mathrm{St}}[s] \tag{C.15}$$

can be estimated using recursive Monte Carlo. Using the radius for a Dirichlet-Neumann problem, we have

$$\begin{aligned} \int_{\partial \mathrm{St}(x,R)} |\rho_\mu(x, y) P^{\mathrm{B}}(x, y)| \, \mathrm{d}y &\leq \int_{\partial \mathrm{St}(x,R)} |\rho_\mu(x, y)| \, \mathrm{d}y \int_{\partial \mathrm{St}(x,R)} |P^{\mathrm{B}}(x, y)| \, \mathrm{d}y \\ &= \int_{\partial \mathrm{St}(x,R)} |\rho_\mu(x, y)| \, \mathrm{d}y, \end{aligned} \tag{C.16}$$

where we used Hölder's inequality and Eq. C.12. To ensure that

$$\int_{\partial \mathrm{St}(x,R)} |\rho_\mu(x, y)| dy \leq 1, \tag{C.17}$$

we follow Sec. 4.3.3 to further restrict R such that $|\rho_\mu(x, y)| <= 1$ for all y. If $|\rho_\mu(x, y)| < 1$ for *any* y, then the inequality becomes strict even before we consider the ε-shell approximation. This explains why WoSt with Robin conditions can also terminate walks using Russian roulette (Sec. 4.3.4), which is not possible when the boundary conditions are Dirichlet and Neumann.

Appendix D

Reflectance & Radius Bound In 2D Domains

To obtain an explicit expression for the reflectance in 2D, we substitute the 2D Green's function and Poisson kernel of a ball $\mathrm{B}(x, R)$ (App. A.1.2) into Eq. 4.7 to obtain

$$\rho_\mu(x, z) = 1 - \frac{\mu(z)\, r}{\cos\theta} \left(\log(R) - \log(r)\right), \tag{D.1}$$

where $r = \|z - x\|$ and $\cos\theta = {}^{n_z \cdot (z-x)}/_{r}$. To restrict $\rho_\mu \in [0, 1]$, we require

$$\frac{\mu(z)\, r}{\cos\theta} \left(\log(R) - \log(r)\right) \leq 1. \tag{D.2}$$

Rearranging terms then gives us an upper bound on the radius R,

$$R \leq r \exp\left(\frac{\cos\theta}{\mu(z)\, r}\right), \tag{D.3}$$

which must hold at *all* points $z \in \partial \mathrm{St}_\mathrm{R}$.

Similar to Sec. 6.2.2, we can compute a tight radius bound for a 2D line segment l using the maximum coefficient $\mu^{\max} := \max(\mu(z))$ for all points $z \in l$, and a distance h from x to the plane l lies on. In particular, letting $r = {}^{h}/_{\cos\theta}$ in Eq. D.3, we have:

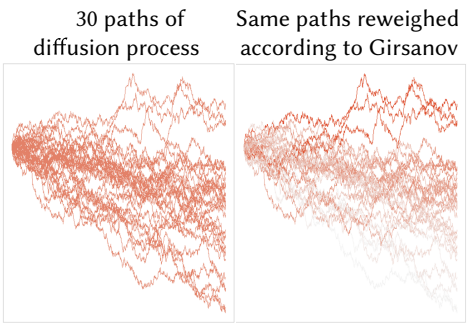
$$R \leq \frac{h}{\cos\theta} \exp\left(\frac{\cos^2\theta}{\mu(z)\, h}\right). \tag{D.4}$$

As before, we minimize this equation with respect to $\cos\theta$. This gives us an analytical expression $\sqrt{{}^{\mu^{\max} h}/_{2}}$ for the cosine, which we clamp between the minimum and maximum cosine values achieved at the closest and farthest points on l (respectively). We then plug the resulting cosine value back into Eq. D.4 to compute the radius bound for l.

Appendix E

Girsanov Transformation

A *Girsanov transformation* allows one stochastic process to be expressed in terms of another. The rough intuition is to imagine that we sample paths from one stochastic process X_t with uniform probability—by assigning a different "weight" to these paths, we can then effectively model a different stochastic process Y_t. In our context this transformation allows us to realize more complex diffusion processes in Ch. 5 via ordinary Brownian motion, which we simulate using walk on spheres.



The idea is well-illustrated by considering the process

$$\mathrm{d}X_t = \vec{\omega}(X_t)\,\mathrm{d}t + \mathrm{d}W_t. \tag{E.1}$$

We can then capture the probability that a pure Brownian process W_t takes the same trajectory as the diffusion process X_t via the *importance sampling weight* [191, Ch. 8]

$$Z(W_t) := \mathrm{e}^{\int_0^t \vec{\omega}(W_s)\cdot \mathrm{d}W_s \,-\, \frac{1}{2}\int_0^t |\vec{\omega}(W_s)|^2\,\mathrm{d}s}. \tag{E.2}$$

The inset shows an example, where samples of a Brownian process are colored according to the probability that they agree with X_t.

E.1 Application To PDEs

The weighting function Z in turn enables us to re-write the Feynman–Kac formula from Eq. 2.31 (with diffusion coefficient $\kappa = 1$) in terms of a Brownian motion W_t. In particular, we now have an equivalent stochastic representation

$$u(x) = \mathbb{E}\left[\mathrm{e}^{-\int_0^\tau \sigma(W_t)dt} Z(W_\tau) g(W_\tau) \;+\; \int_0^\tau \mathrm{e}^{-\int_0^t \sigma(W_t)\,\mathrm{d}s} Z(W_t) f(W_t)\,\mathrm{d}t\right]. \tag{E.3}$$

This version of Feynman–Kac can then be used to numerically estimate the solution to the PDE

$$\begin{aligned} \tfrac{1}{2}\Delta u(x) + \vec{\omega}(x)\cdot\nabla u(x) - \sigma(x)u(x) &= -f(x) \quad \text{on } \Omega, \\ u(x) &= g(x) \quad \text{on } \partial\Omega_{\mathrm{D}}, \end{aligned} \tag{E.4}$$

but now using pure Brownian motion without drift—see Sec. 5.2 for further discussion.

E.2 Chain Rule Of Stochastic Calculus

The term $\int_0^t \vec{\omega}(W_s)\cdot \mathrm{d}W_s$ in Eq. E.2 is called a *stochastic integral*, as it is defined with respect to variations of a Brownian process W_s (see Øksendal [191, Ch. 4] for a formal definition). To

evaluate this integral, we turn to *Itô's lemma*, which is the stochastic counterpart of the chain rule: given a twice differential function $\gamma(x) : \mathbb{R}^n \mapsto \mathbb{R}$, Itô's lemma says that the differential $\mathrm{d}\gamma$, as a function of a Brownian process W_s, is given by

$$\mathrm{d}\gamma(W_s) = \nabla\gamma(W_s) \cdot \mathrm{d}W_s + \frac{1}{2}\Delta\gamma(W_s)\,\mathrm{d}s. \tag{E.5}$$

Integrating over time and rearranging terms then yields

$$\int_0^t \nabla\gamma(W_s) \cdot \mathrm{d}W_s = \gamma(W_t) - \gamma(W_0) - \int_0^t \tfrac{1}{2}\Delta\gamma(W_s)\,\mathrm{d}s. \tag{E.6}$$

This integrated version of Itô's lemma allows us to re-express the importance weight Z without a stochastic integral. In particular, if the drift coefficient takes the form $\vec{\omega}(x) = \nabla\gamma(x)$ for any scalar field γ, then an alternative expression for Z is given by

$$Z(W_t) = \mathrm{e}^{\gamma(W_t) \,-\, \gamma(W_0) \,-\, \frac{1}{2}\int_0^t \left(\Delta\gamma(W_s) \,+\, |\nabla\gamma(W_s)|^2\right)\mathrm{d}s}. \tag{E.7}$$

E.3 Derivation Of Eq. 5.3

With this new expression for Z, Eq. E.3 now takes the form

$$u(x) = \mathrm{e}^{-\gamma(x)}\,\mathbb{E}\left[\mathrm{e}^{-\int_0^\tau \sigma'(W_t)\,\mathrm{d}t} g'(W_\tau) \;+\; \int_0^\tau \mathrm{e}^{-\int_0^t \sigma'(W_s)\,\mathrm{d}s} f'(W_t)\,\mathrm{d}t\right], \tag{E.8}$$

where

$$\begin{gathered} W_0 = x, \qquad f'(x) \coloneqq \mathrm{e}^{\gamma(x)} f(x), \qquad g'(x) \coloneqq \mathrm{e}^{\gamma(x)} g(x), \\ \text{and} \quad \sigma'(x) \coloneqq \sigma(x) + \frac{1}{2}\left(\Delta\gamma(x) + |\nabla\gamma(x)|^2\right). \end{gathered}$$

Making the substitution $U(x) \coloneqq \mathrm{e}^{\gamma(x)} u(x)$, Eq. E.8 then provides the solution to the following PDE without a 1st order drift term:

$$\begin{aligned} \tfrac{1}{2}\Delta U(x) - \sigma'(x) U(x) &= -f(x) \quad \text{on } \Omega, \\ U(x) &= g(x) \quad \text{on } \partial\Omega_{\mathrm{D}}. \end{aligned} \tag{E.9}$$

This PDE is equivalent to Eq. E.4, and we obtain the PDE transformation from Eq. 5.2 to Eq. 5.3 by letting $\gamma(x) = \frac{1}{2}\ln(\kappa(x))$.

Appendix F

Pseudocode: Accelerated Geometric Queries For Walk On Stars

Here we provide pseudocode for the geometric queries described in Sec. 6.2.

ALGORITHM 8: StarRegionRadius$(t,\ x,\ R^{\max} \leftarrow \infty)$
Note: Code annotated with comments in gray, blue and green corresponds to handling of Dirichlet, Neumann and Robin conditions (respectively).

Input: Triangle t with Robin coefficients $\mu^{\min}$ and $\mu^{\max}$, query point $x \in \mathbb{R}^3$, radius bound $R^{\max}$.
Output: Radius of star-shaped region for triangle t.

$R_t \leftarrow R^{\max}$ ▷*Initialize radius value for triangle t*
$d_t^c,\ x_t^c \leftarrow$ ClosestPt$(t,\ x)$ ▷*Compute closest pt x_t^c on t from x [62, Sec. 5.1.5]*
if $d_t^c > R_t$ **then return** R_t ▷*t is outside radius bound, return the bound*
if $t.\mu^{\max} \equiv \infty$ **then return** d_t^c ▷*t has Dirichlet conditions, return distance to x_t^c*
$n_t \leftarrow$ Normal(t)
for e **in** t.adjacentEdges **do** ▷*Compute distance to closest silhouette edge adjacent to t*
    $p_e \leftarrow$ ClosestPt$(e,\ x)$
    $v \leftarrow p_e - x$
    $d_e \leftarrow \|v\|$
    if $d_e < R_t$ **then**
        hasAdjacentTriangle, $n_{\text{adj}} \leftarrow$ AdjacentNormal$(t,\ e)$
        isSilhouetteEdge $\leftarrow$ **not** hasAdjacentTriangle **or** $(v \cdot n_t) \cdot (v \cdot n_{\text{adj}}) \leq 0$
        if isSilhouetteEdge **then** $R_t \leftarrow \min(R_t,\ d_e)$
if $t.\mu^{\min} \equiv 0$ **then return** R_t ▷*t has Neumann conditions, return distance to closest silhouette edge*
else ▷*t has Robin conditions, compute radius bound for t (Eq. 6.2)*
    $d_t^f,\ x_t^f \leftarrow$ FarthestPt$(t,\ x)$
    $\cos^{\max}\theta \leftarrow |n_t \cdot (x - x_t^f)| \ /\ d_t^f$
    $\cos^{\min}\theta \leftarrow |n_t \cdot (x - x_t^c)| \ /\ d_t^c$
    $h \leftarrow$ DistancePlane$(x_t^c,\ n_t)$
    $\mu h \leftarrow t.\mu^{\max} \cdot h$
    if $\sqrt{\mu h} < \cos^{\min}\theta$ **then return** R_t
    $\cos\theta \leftarrow$ Clamp$(\sqrt{\mu h\ /\ 3},\ \cos^{\min}\theta,\ \cos^{\max}\theta)$
    return $\min\left(R_t,\ \frac{\mu h^2}{\mu h \cos\theta\ -\ \cos^3\theta}\right)$ ▷*Return computed radius bound*

ALGORITHM 9: StarRegionRadius($T \leftarrow \text{SNCH}(\partial\Omega_R),\ x,\ R,\ d_T^{\min} \leftarrow 0$)

Input: Spatialized normal cone hierarchy T, query point $x \in \mathbb{R}^3$, current estimate R of radius, minimum distance $d_T^{\min}$ to T's aabb from x ($d_T^{\min} = 0$ when $x \in T.\text{aabb}$).
Output: Radius of star-shaped region containing subset of reflecting boundary $\partial\Omega_R$.

if $d_T^{\min} > R$ **then return** R ▷*Ignore current SNCH node if it is further than R*
if T.isLeaf **then**
  for t **in** T.triangles **do**
    $R \leftarrow \text{StarRegionRadius}(t,\ x,\ R)$ ▷*Compute radius bound for triangle t (Alg. 8)*
else
  visitLeft, $R_{\text{left}}^{\min}, R_{\text{left}}^{\max} \leftarrow \text{VisitNode}(T.\text{left}, x, R)$ ▷*Test whether to visit left node (Alg. 10)*
  if visitLeft **then** $R \leftarrow \min(R,\ R_{\text{left}}^{\max})$
  visitRight, $R_{\text{right}}^{\min}, R_{\text{right}}^{\max} \leftarrow \text{VisitNode}(T.\text{right}, x, R)$ ▷*Test whether to visit right node (Alg. 10)*
  if visitRight **then** $R \leftarrow \min(R,\ R_{\text{right}}^{\max})$
  if visitLeft **and** visitRight **then**
    if $R_{\text{left}}^{\min} < R_{\text{right}}^{\min}$ **then** ▷*Visit closer node first*
      $R \leftarrow \text{StarRegionRadius}(T.\text{left},\ x,\ R,\ R_{\text{left}}^{\min})$
      $R \leftarrow \text{StarRegionRadius}(T.\text{right},\ x,\ R,\ R_{\text{right}}^{\min})$
    else
      $R \leftarrow \text{StarRegionRadius}(T.\text{right},\ x,\ R,\ R_{\text{right}}^{\min})$
      $R \leftarrow \text{StarRegionRadius}(T.\text{left},\ x,\ R,\ R_{\text{left}}^{\min})$
  else if visitLeft **then** $R \leftarrow \text{StarRegionRadius}(T.\text{left},\ x,\ R,\ R_{\text{left}}^{\min})$ ▷*Visit left node*
  else if visitRight **then** $R \leftarrow \text{StarRegionRadius}(T.\text{right},\ x,\ R,\ R_{\text{right}}^{\min})$ ▷*Visit right node*
return R

ALGORITHM 10: VisitNode($T,\ x,\ R \leftarrow \infty$)

Note: Code annotated with comments in gray, blue and green corresponds to handling of Dirichlet, Neumann and Robin conditions (respectively).

Input: Spatialized normal cone hierarchy node T with Robin coefficients $\mu^{\min}$ and $\mu^{\max}$, query point $x \in \mathbb{R}^3$, current estimate R of radius.
Output: Whether to traverse the node, with min and max bounds for the radius.

visit, $d^{\min}, d^{\max} \leftarrow \text{Intersect}(T.\text{aabb}, x, R)$ ▷*Check if T's aabb intersects ball* $\text{B}(x, R)$
if not visit **then return** false, $0, \infty$ ▷*Skip node as aabb does not intersect* B
if $T.\mu^{\min} \equiv \infty$ **then return** true, $d^{\min}, d^{\max}$ ▷*T contains only Dirichlet boundary, visit node*
visit, $|\cos^{\min}\theta|, |\cos^{\max}\theta| \leftarrow \text{HasSilhouette}(T.\text{aabb}, T.\text{cone}, x)$ ▷*Sawhney et al. [224, Alg. 4]*
if visit **then return** true, $d^{\min}, \infty$ ▷*Normal & view cones possibly contain orthogonal directions*
if $T.\mu^{\max} \equiv 0$ **then return** false, $0, \infty$ ▷*Neumann boundary in T is front-/back-facing, skip node*
if $d^{\max} \leq |\cos^{\min}\theta|\ /\ T.\mu^{\max}$ **then** $R^{\min} \leftarrow \infty$ ▷*No min radius bound for Robin boundary in T*
else $R^{\min} \leftarrow d^{\min} \Big/ \left(1 - \frac{|\cos^{\min}\theta|}{T.\mu^{\max} \cdot d^{\max}}\right)$ ▷*Compute min radius bound (Eq. 4.11)*
if $d^{\min} \leq |\cos^{\max}\theta|\ /\ T.\mu^{\min}$ **then** $R^{\max} \leftarrow \infty$ ▷*No max radius bound for Robin boundary in T*
else $R^{\max} \leftarrow d^{\max} \Big/ \left(1 - \frac{|\cos^{\max}\theta|}{T.\mu^{\min} \cdot d^{\min}}\right)$ ▷*Compute max radius bound (Eq. 4.11)*
return true, $R^{\min}, R^{\max}$ ▷*Visit node*

ALGORITHM 11: PointSamplingQuery$(T,\ x,\ R)$

Input: Binary tree T, and sphere $\partial\mathrm{B}(x, R)$ with center $x \in \mathbb{R}^3$ and radius R.

Output: Point sample $z \in \partial\Omega_{\mathrm{R}}$, unit outward normal n_z, and pdf_z. It is not guaranteed $z \in \partial\mathrm{B}(x, R)$, but z is likely to be close to x. No sample is returned if $\partial\mathrm{B}$ is empty.

$t,\ \mathrm{pdf}_t \leftarrow$ SampleTriangle$(T,\ x,\ R)$ ▷*Sample triangle t inside or intersecting with ∂B (Alg. 12)*

if t **not** NULL **then**

  $z,\ n_z,\ \mathrm{pdf}_z \leftarrow$ SamplePoint(t) ▷*Sample point z uniformly on t*

  $\mathrm{pdf}_z \leftarrow \mathrm{pdf}_z \cdot \mathrm{pdf}_t$

  return $z,\ n_z,\ \mathrm{pdf}_z$

return NULL, NULL, 0

ALGORITHM 12: SampleTriangle$(T,\ x,\ R,\ t \leftarrow \mathrm{NULL},\ \mathrm{pdf}_t \leftarrow 0,\ \mathrm{pdf}_T \leftarrow 1)$

Input: Binary tree T, sphere $\partial\mathrm{B}(x, R)$ with center $x \in \mathbb{R}^3$ and radius R, triangle t to be selected and its sampling pdf_t, and the probability pdf_T of traversing a random branch in T.

Output: Randomly selected triangle t inside or intersecting with $\partial\mathrm{B}(x, R)$, and its sampling pdf_t. No triangle is selected if ∂B is empty.

if T.isLeaf **then**

  totalArea $\leftarrow 0$

  for t_T **in** T.triangles **do** ▷*Use weighted importance resampling (Sec. 3.1.2) to select triangle*

    if Intersect$(t_T,\ x,\ R)$ **then** ▷*Check if triangle t_T is inside or intersects with $\partial\mathrm{B}(x, R)$*

      totalArea $\leftarrow$ totalArea $+$ Area(t_T)

      if Rand$() \cdot$ totalArea $<$ Area(t_T) **then**

        $t \leftarrow t_T$

        $\mathrm{pdf}_t \leftarrow \mathrm{pdf}_T \cdot$ Area(t)

  if totalArea > 0 **then** $\mathrm{pdf}_t \leftarrow \mathrm{pdf}_t$ / totalArea

else

  $w_{\mathrm{left}} \leftarrow$ Intersect$(T$.left.aabb$,\ x,\ R)$?

    $G^{\mathbb{R}^3}(x,$ Centroid$(T$.left.aabb$))\ :\ 0$ ▷*Weight left subtree by proximity to x*

  $w_{\mathrm{right}} \leftarrow$ Intersect$(T$.right.aabb$,\ x,\ R)$?

    $G^{\mathbb{R}^3}(x,$ Centroid$(T$.right.aabb$))\ :\ 0$ ▷*Weight right subtree by proximity to x*

  $W \leftarrow w_{\mathrm{left}} + w_{\mathrm{right}}$

  if $W > 0$ **then**

    $\mathbb{P}_{\mathrm{left}} \leftarrow w_{\mathrm{left}}$ / W ▷*Compute probability of traversing left subtree*

    if Rand$() < \mathbb{P}_{\mathrm{left}}$ **then**

      $\mathrm{pdf}_{\mathrm{left}} \leftarrow \mathrm{pdf}_T \cdot \mathbb{P}_{\mathrm{left}}$

      $t,\ \mathrm{pdf}_t \leftarrow$ SampleTriangle$(T$.left$,\ x,\ R,\ t,\ \mathrm{pdf}_t,\ \mathrm{pdf}_{\mathrm{left}})$ ▷*Traverse left subtree*

    else

      $\mathrm{pdf}_{\mathrm{right}} \leftarrow \mathrm{pdf}_T \cdot (1 - \mathbb{P}_{\mathrm{left}})$

      $t,\ \mathrm{pdf}_t \leftarrow$ SampleTriangle$(T$.right$,\ x,\ R,\ t,\ \mathrm{pdf}_t,\ \mathrm{pdf}_{\mathrm{right}})$ ▷*Traverse right subtree*

return $t,\ \mathrm{pdf}_t$