Walkin' Robin: Walk on Stars with Robin Boundary Conditions

BAILEY MILLER*, Carnegie Mellon University, USA
ROHAN SAWHNEY*, NVIDIA, USA
KEENAN CRANE†, Carnegie Mellon University, USA
IOANNIS GKIOULEKAS†, Carnegie Mellon University, USA

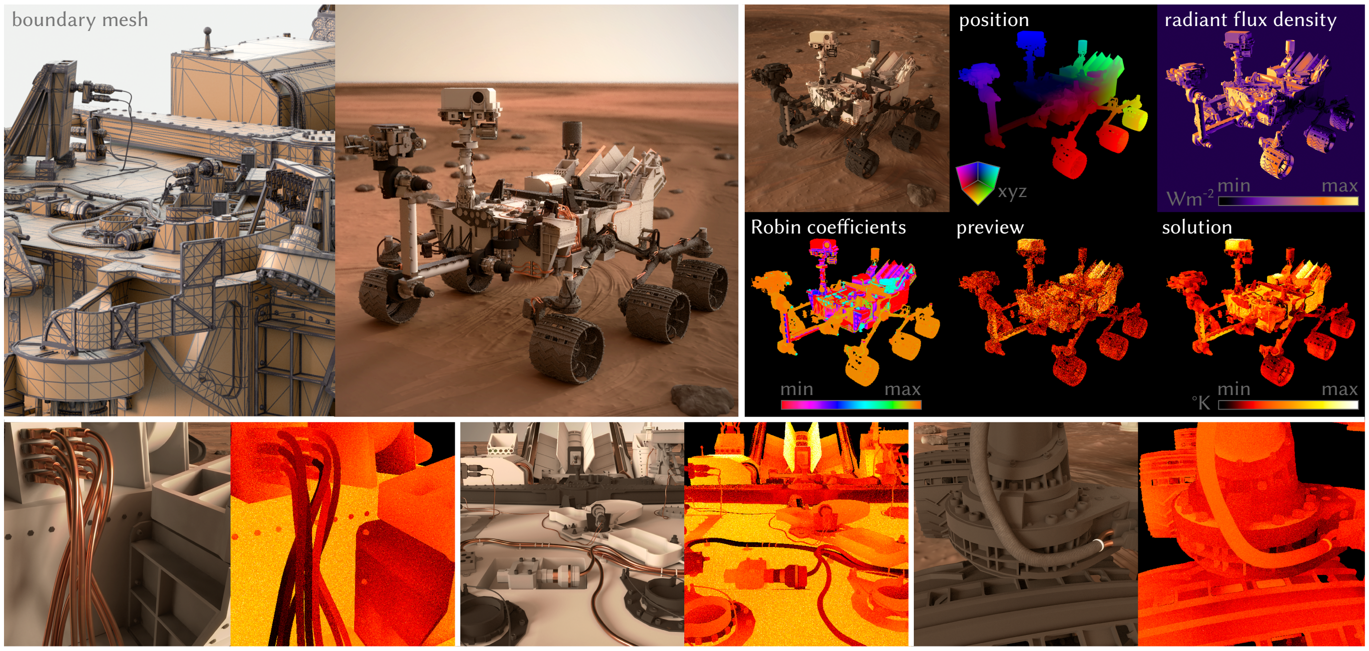


Figure 1. Thermal analysis of NASA's *Curiosity* Mars rover, mocked up on an artist-generated model not meant for simulation *(top left)*. Keeping temperatures within specified thermal limits is critical to mission success—but thermal modeling is historically difficult to integrate into the design phase, due to intricate geometry not easily captured via finite element models (Figure 4). A grid-free Monte Carlo solver that supports Robin boundary conditions enables us to compute realistic temperature estimates quickly and progressively even for extremely complex geometry, without needing to volumetrically mesh the domain. Here 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)*.

Numerous scientific and engineering applications require solutions to boundary value problems (BVPs) involving elliptic partial differential equations, such as the Laplace or Poisson equations, on geometrically intricate domains. We develop a Monte Carlo method for solving such BVPs with arbitrary first-order linear boundary conditions—Dirichlet, Neumann, and Robin. Our method directly generalizes the *walk on stars (WoSt)* algorithm, which previously tackled only the first two types of boundary conditions, with a few simple modifications. Unlike conventional numerical methods, WoSt does not need finite element meshing or global solves. Similar to Monte Carlo rendering, it instead computes pointwise solution estimates by simulating random walks along star-shaped regions inside the BVP domain, using efficient ray-intersection and distance queries. To ensure WoSt produces *bounded-variance* estimates in the presence of Robin boundary conditions, we show that it is sufficient to modify how WoSt selects the size of these star-shaped regions. Our generalized WoSt algorithm reduces estimation error by orders of magnitude relative to alternative grid-free methods such as the *walk on boundary* algorithm. We also develop *bidirectional* and *boundary value caching* strategies to further reduce estimation error. Our algorithm is trivial to parallelize, scales sublinearly with increasing geometric detail, and enables progressive and view-dependent evaluation.

*and † indicate equal contribution.

Authors' addresses: Bailey Miller, bmmiller@andrew.cmu.edu, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA; Rohan Sawhney, rsawhney@nvidia.com, NVIDIA, 2788 San Tomas Expy, Santa Clara, CA, 95051, USA; Keenan Crane, kmcrane@cs.cmu.edu, Carnegie Mellon University, USA; Ioannis Gkioulekas, igkioule@cs.cmu.edu, Carnegie Mellon University, USA.

CCS Concepts: • **Mathematics of computing** → **Partial differential equations**; **Integral equations**; **Probabilistic algorithms**.

Additional Key Words and Phrases: Partial differential equations, Monte Carlo methods, walk on spheres

ACM Reference Format:
Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2024. Walkin' Robin: Walk on Stars with Robin Boundary Conditions. *ACM Trans. Graph.* 43, 4, Article 41 (July 2024), 18 pages. https://doi.org/10.1145/3658153

1 INTRODUCTION

With the rapid increase in the complexity of geometric models, grid-free Monte Carlo methods for boundary value problems (BVPs), such as *walk on spheres (WoS)* [Muller 1956], have received renewed interest from the scientific computing and computer graphics communities [Mascagni and Simonov 2004; Juba et al. 2016; Sawhney and Crane 2020]. WoS is attractive as it does not require a volumetric mesh of the problem domain, nor a high-quality mesh of its boundary. Similar to Monte Carlo rendering, WoS is also output-sensitive, parallelizes easily, and scales well with model complexity without the need for any geometric preprocessing. These computational features have encouraged researchers to use WoS in recent years to solve a broad set of steady-state, time-dependent and even weakly non-linear partial differential equations (PDEs) [Bossy et al. 2010; Kyprianou et al. 2017; Nabizadeh et al. 2021; Sawhney et al. 2022; Rioux-Lavoie et al. 2022; Bati et al. 2023; De Lambilly et al. 2023]. Yet, despite handling complex boundary shapes, WoS does not support boundary conditions beyond simple Dirichlet ones.

Boundary conditions control the BVP solution by imposing functional constraints on its values at boundary points—these constraints correspond to application-dependent constraints on physical systems, controlling their temperature, voltage, force, velocity, and so on. The *walk on stars (WoSt)* algorithm [Sawhney et al. 2023] recently generalized WoS to support BVPs with mixed *Dirichlet* and *Neumann* conditions, which constrain the solution and its derivatives (*resp.*) on the boundary. As Figure 2 (*top row*) shows, WoSt solves such BVPs by simulating random walks along *star-shaped regions* inside the problem domain—these walks model *Brownian motion*, which is absorbed into Dirichlet boundaries and reflected off Neumann boundaries [Øksendal 2003; Grebenkov 2006, 2007].

Our goal in this paper is to further extend the types of boundary conditions WoSt supports, by also handling *Robin* conditions—corresponding, from the Brownian motion viewpoint, to boundaries that are both reflecting and absorbing (Figure 3). Robin conditions provide greater physical realism than Dirichlet or Neumann conditions when modeling real-world thermal, electromagnetic, elastic, and fluidic materials: most real materials are both absorbing and reflecting, rather than purely absorbing (*e.g.*, a black body) or purely reflecting (*e.g.*, a perfect insulator). Additionally, Monte Carlo estimators are typically more efficient in Robin-dominated than Neumann-dominated problems (Section 4): random walks simulating *partially reflecting Brownian motion* can be absorbed on $\partial\Omega_R$ [Grebenkov 2006], whereas reflecting Neumann walks must take many steps to reach $\partial\Omega_D$, resulting in high computation time.

In particular, we generalize WoSt to solve BVPs of the form:

$$\begin{aligned} \Delta u(x) &= f(x) && \text{on } \Omega, \\ u(x) &= g(x) && \text{on } \partial\Omega_D, \\ \frac{\partial u(x)}{\partial n_x} - \mu(x)u(x) &= h(x) && \text{on } \partial\Omega_R, \end{aligned} \tag{1}$$

where the boundary of the domain $\Omega \subset \mathbb{R}^N$ is partitioned into a Dirichlet part $\partial\Omega_D$ and a Robin part $\partial\Omega_R$ with prescribed values g and h (*resp.*). Here Δ is the negative-semidefinite Laplacian, f is a given source term, n_x is the unit outward normal to $\partial\Omega_R$ at x, and $\mu \in \mathbb{R}_{\geq 0}$ is a non-negative Robin coefficient that can vary over $\partial\Omega_R$ (negative μ values model emissive boundaries, which we do not consider in this paper). We recover Neumann conditions when $\mu = 0$, and Dirichlet conditions as $\mu \longrightarrow \infty$. For algorithmic convenience we treat pure Dirichlet conditions (*i.e.*, $u = g$) as separate from Robin, and focus on $N = 2$ and 3 in this paper.

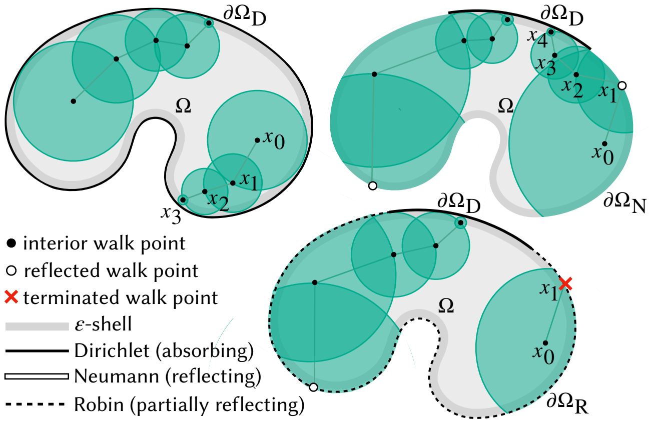
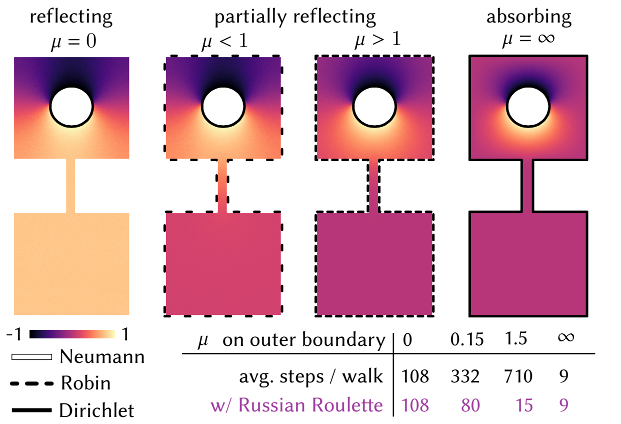


Figure 2. We solve BVPs with arbitrary first-order linear boundary conditions (Dirichlet, Neumann, Robin) using walk on stars. *Top left:* With a purely absorbing Dirichlet boundary $\partial\Omega_D$, WoSt is equivalent to WoS, which repeatedly jumps to a random point on the largest sphere around the current walk location, and terminates when the walk enters an ε-shell $\partial\Omega_D^\varepsilon$ around the boundary. *Top right:* With a reflecting Neumann boundary $\partial\Omega_N$, WoSt replaces spheres with star-shaped regions—formed using spheres that can contain a subset of $\partial\Omega_N$—and selects the next walk location by intersecting a random ray direction with the current star-shaped region. *Bottom:* With a partially-reflecting Robin boundary $\partial\Omega_R$, WoSt still uses star-shaped regions, but can additionally terminate walks on the Robin boundary.

We show that this generalization requires changing only how WoSt selects the size of star-shaped regions (Figure 5), to ensure its estimates have bounded variance. To this end, we define a *reflectance function* (Equation 7), which depends on the Robin coefficient μ and serves as a multiplicative weight for boundary contributions along each step of a random walk. Then, at each step, we select the size of the star-shaped region so as to bound the reflectance function between 0 and 1, and thus ensure that walk contributions remain positive and do not grow uncontrollably with walk length. Bounding the reflectance value also allows us to improve efficiency, by enabling the use of *Russian roulette* [Pharr et al. 2016, Section 13.7] to probabilistically terminate walks on $\partial\Omega_R$ (Figures 2 and 3). The resulting estimator has guaranteed Monte Carlo convergence with increasing walk count for *any* combination of Dirichlet, Neumann and Robin conditions (Figures 11 and 12). It also improves estimation error by orders of magnitude compared to other grid-free techniques for the BVP in Equation 1, such as the *walk on boundary (WoB)* algorithm [Sabelfeld and Simonov 2013; Sugimoto et al. 2023].

From an implementation perspective, our approach requires only small modifications (Algorithm 1) to the original WoSt algorithm (Sawhney et al. [2023, Algorithm 1]), to account for the reflectance-dependent size of star-shaped regions. In particular, we show how to efficiently determine region size using the same *spatialized normal cone hierarchy (SNCH)* [Johnson and Cohen 2001] as Sawhney et al.



μ on outer boundary	0	0.15	1.5	∞
avg. steps / walk	108	332	710	9
w/ Russian Roulette	108	80	15	9

Figure 3. A non-negative Robin coefficient μ linearly interpolates between absorbing Dirichlet ($\mu = \infty$) and reflecting Neumann ($\mu = 0$) conditions, which prescribe boundary values and derivatives (*resp.*) to a PDE (here a Laplace equation). As μ increases and the boundary becomes less reflecting, we can apply *Russian roulette* to probabilistically terminate random walks on the Robin boundary, and reduce the average number of steps per walk.

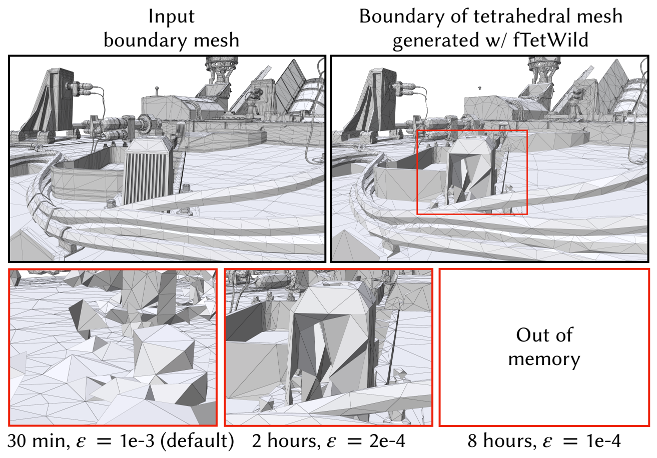

Input
boundary mesh
Boundary of tetrahedral mesh
generated w/ fTetWild
Out of
memory
30 min, ε = 1e-3 (default)
2 hours, ε = 2e-4
8 hours, ε = 1e-4

Figure 4. Generating tetrahedral meshes for accurate FEM simulation can be challenging, as state-of-the-art 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 [Hu et al. 2020] on the Mars Rover from Figure 1 using an AMD Ryzen Threadripper PRO with 64 GB RAM. TetGen [Si 2015] cannot tetrahedralize this model as it contains self-intersections.

[2023], which visits only a small fraction of the primitives on the boundary of the BVP domain.

Finally, we extend existing variance reduction techniques for WoSt, such as *bidirectional* formulations [Qi et al. 2022] and *boundary value caching (BVC)* [Miller et al. 2023], to Robin problems to reduce the noise in solution estimates output by our generalized WoSt algorithm (Figure 10). Overall, our algorithm significantly expands the scope of BVPs that can be tackled compared to the original WoS algorithm by Muller [1956], while retaining its computational benefits over conventional grid-based solvers, such as geometric robustness and scalability, output sensitivity and trivial parallelism.

2 RELATED WORK

A large variety of numerical methods have been developed to solve BVPs, given their central importance in science and engineering. We discuss two broad categories: traditional grid-based deterministic methods, and emerging grid-free Monte Carlo methods like ours. We refer to Sawhney and Crane [2020, Section 7] and Sawhney et al. [2022, Section 7] for a thorough evaluation of tradeoffs.

2.1 Grid-based PDE Solvers

Grid-based methods require spatial discretization of the domain (meshing or sampling), and solving of globally coupled systems of equations defined on such a discretization. These requirements makes grid-based methods difficult to parallelize and prone to aliasing in the geometry, boundary conditions, source terms, and solution [Sawhney and Crane 2020; Sawhney et al. 2022, 2023, Figures 27, 22 & 4]. The primary bottleneck with the *finite element method (FEM)* is often not the solve itself, but rather the cost of robustly meshing large, detailed, and imperfect geometry (*e.g.*, with self-intersections) that even state-of-the-art methods [Si 2015; Hu et al. 2020] struggle with (*e.g.*, Figure 4 and Sawhney et al. [2023, Figure 5]). In a process akin to meshing, "meshless" FEM must sample the *entire* domain [Li and Liu 2007]—convergence can stagnate due to poor sampling [Flyer et al. 2016], or completely fail without problem-specific tuning of parameters [Sawhney et al. 2022, Figures 24 & 25]. The *boundary element method (BEM)* must discretize the domain boundary, then solve a *dense* linear system that grows *quadratically* with geometric detail. Consequently, BEM requires special matrix approximation schemes to scale to large geometries [Hackbusch 2015], and must be coupled with FEM or finite differences (FD) to handle volumetric inputs such as source terms and PDE coefficients [Coleman et al. 1991; Partridge et al. 2012].

FD methods are attractive due to their simplicity of implementation, requiring only a regular grid or octree [Losasso et al. 2006]. However, they typically necessitate significant grid refinement to avoid aliasing, resulting in compute and memory complexity that scales cubicly with resolution. Additionally, as the grid cells are axis aligned, such methods make enforcement of boundary conditions difficult. Recent learning-based methods such as *physics-informed neural networks (PINNs)* [Raissi et al. 2019] can sidestep cubic complexity issues, by replacing grids with coordinate neural networks that enable dense evaluation in a resolution-independent manner. However, PINNs require expensive training over the entire domain (analogous to a global solve), careful selection of training samples, and cumbersome hyperparameter tuning. They also struggle with accurately enforcing boundary conditions, especially in complex geometric domains [Krishnapriyan et al. 2021].

In contrast, Monte Carlo methods such as WoSt [Sawhney et al. 2023] do not require global solves or training; they can instead evaluate PDE solutions independently at points of interest in an *embarrassingly parallel* manner. They do not require a background grid to discretize problem inputs, and hence also avoid aliasing. They instead query geometry using acceleration structures (*e.g.*, an SNCH) derived from *bounding volume hierarchies (BVH)*: such structures use little memory, can be built in a fraction of the time needed for meshing or training [Sawhney and Crane 2020, Figure 25], and preserve sharp edges, small details, and thin features in the geometry *exactly*. The outputs of Monte Carlo methods, though noisy, improve progressively (Figure 1) with more samples *if walk throughput is bounded*, thus providing greater robustness compared to the outputs of grid-based methods, which can be irreparably corrupted because of a single bad discretization element.

2.2 Grid-free Monte Carlo Methods

Monte Carlo methods for BVPs simulate random walks modeling continuous stochastic processes (such as Brownian motion) that in aggregate solve a large class of elliptic and parabolic PDEs—we refer to Sawhney et al. [2022, Section 2] and Øksendal [2003, Chapters 8 & 9] for key results and derivations. Unlike grid-based methods, Monte Carlo methods do not require domain or boundary discretization, and support independent pointwise solution evaluation.

2.2.1 Discretized Random Walks. Classical Monte Carlo methods simulate random walks with time stepping, similar to ray marching in rendering. These methods correspond to integration schemes for stochastic differential equations [Morillon 1997; Higham 2001], and are generally not well-suited to solving BVPs with complex domains, as time discretization requires trading off between slow runtimes due to many small steps in a walk, and high bias due to inaccurate large steps [Sawhney et al. 2022, 2023, Figures 28 & 16].

2.2.2 Continuous Random Walks. Discretization-free methods, starting with WoS [Muller 1956; Sawhney and Crane 2020], use closed-form distributions to *exactly* model large steps of a Brownian motion, and incur only a small bias on the boundary due to approximate walk termination (Figures 6 and 11). Hence, they generally have a much more favorable runtime-to-bias tradeoff compared to discretization-based or hybrid methods [Zhou and Cai 2016; Simonov 2017]. Methods that build on WoS have also improved significantly in recent years, through optimized implementations [Krayer and Müller 2021; Mossberg 2021], variance reduction techniques [Nabizadeh et al. 2021; Qi et al. 2022; Miller et al. 2023; Bakbouk and Peers 2023; Li et al. 2023], and extensions to more general BVPs [Sawhney et al. 2022; Rioux-Lavoie et al. 2022; Yılmazer et al. 2022; Sawhney et al. 2023; Bati et al. 2023; De Lambilly et al. 2023]. We continue this trend by extending the WoSt algorithm, along with some of these variance reduction techniques, to BVPs with Robin boundary conditions. Crucially, our WoSt algorithm maintains the advantages of WoS over grid-based methods by exactly modeling large steps of a partially reflecting Brownian motion.

Lastly, the *walk on boundary (WoB)* algorithm [Sabelfeld and Simonov 2013; Sugimoto et al. 2023] is an alternative discretization-free method. It solves BVPs by using random walks that jump between different locations on the domain boundary, to recursively evaluate single and double layer potentials at those locations. As we show in Figure 12, WoB currently suffers from very high variance and bias in *any* non-convex domain with Dirichlet or Neumann conditions, with Robin conditions further exacerbating these issues. Fundamentally, these issues stem from the fact that in non-convex domains, walk contributions in WoB grow uncontrollably with walk length, resulting in solution estimates that have unbounded variance. This problem necessitates artificially truncating walk length to curtail variance, but unfortunately this approach results in severe estimation bias—Section 7.2 provides further details. In contrast, our WoSt algorithm provides estimates with bounded variance that decreases at the standard Monte Carlo rate of convergence in all domains. Furthermore, unlike WoB, it allows for *unbiased* and low-variance early walk termination by applying Russian roulette on reflectance values (Section 4.3).

3 BACKGROUND

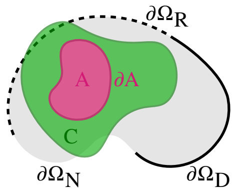
We review two topics that serve as the basis of our method: formulating the solution to BVPs such as Equation 1 as a *boundary integral equation (BIE)* (Section 3.1), and using WoSt to recursively evaluate BIEs through Monte Carlo integration (Section 3.2). Sawhney et al. [2023, Sections 3 & 4] provide a detailed discussion of these topics.

3.1 Boundary Integral Equation

We assume Ω is a watertight domain with smooth boundary $\partial\Omega$, and A and C are arbitrary subsets of Ω and $\mathbb{R}^N$ (*resp.*). Then for any point $x \in \mathbb{R}^N$, the solution to a Poisson equation satisfies [Costabel 1987; Hunter and Pullan 2001]

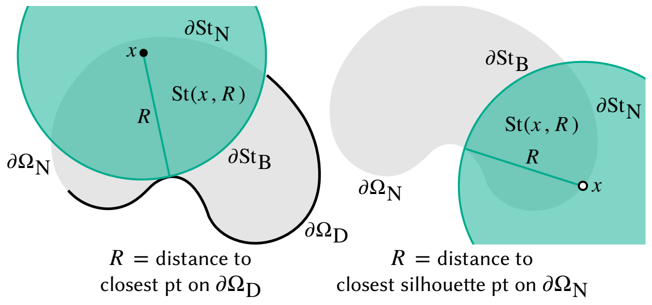
$$\alpha(x)\, u(x) = \int_{\partial \mathrm{A}} P^{\mathrm{C}}(x,z)\, u(z) \;-\; G^{\mathrm{C}}(x,z)\, \frac{\partial u(z)}{\partial n_z}\, \mathrm{d}z + \int_{\mathrm{A}} G^{\mathrm{C}}(x,y)\, f(y)\, \mathrm{d}y, \tag{2}$$

where $\alpha(x) = 1$ if $x \in \mathrm{A}$, $1/2$ if $x \in \partial\mathrm{A}$, and 0 otherwise. The *Green's function* G^C and *Poisson kernel* $P^C := \partial G^C/\partial n$ for a Poisson equation are typically not known in closed form for arbitrary C, but explicit expressions are available for important special cases, *e.g.*, free space ($C = \mathbb{R}^N$) and the ball ($C = \mathrm{B}$) [Sawhney et al. 2023, Appendix A.1].



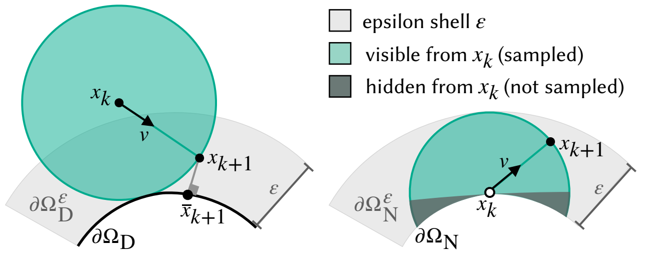
To use Equation 2, we must determine solution values $u(z)$ and their normal derivatives $\partial u/\partial n_z(z)$ on the boundary $\partial\mathrm{A}$ (inset). The only information we have available about these functions is through the various boundary conditions on disjoint parts of $\partial\Omega$: Dirichlet conditions prescribe solution values $u = g$ on $\partial\Omega_{\mathrm{D}}$, Neumann conditions prescribe derivatives $\partial u/\partial n = h$ on $\partial\Omega_{\mathrm{N}}$, and Robin conditions prescribe a linear combination of the two, $\partial u/\partial n - \mu\, u = h$, on $\partial\Omega_{\mathrm{R}}$; the source term f is specified everywhere inside Ω (and hence A). In general though, u and $\partial u/\partial n$ are not known on $\partial\mathrm{A}$, so we must estimate them using a numerical method such as WoSt (Section 3.2).

Though we focus on the Poisson equation for simplicity, the BIE extends directly to the *screened Poisson equation* $\Delta u - \sigma u = f$ with absorption coefficient $\sigma \in \mathbb{R}_{\geq 0}$: the only modification to Equation 2 is to replace the Green's function and Poisson kernel with their screened counterparts [Sawhney et al. 2023, Appendix A.2]. The BIE also applies, in principle, to a much broader class of linear elliptic PDEs with variable diffusion, drift, and absorption coefficients [Sawhney et al. 2022]—Section 8 provides further discussion. Lastly, Sawhney et al. [2023, Appendix B] describe how to generalize the BIE to open domains and double-sided boundary conditions; no special treatment is needed to extend Robin conditions to this setting.



$\partial\mathrm{St}_\mathrm{N}$
x
$\mathrm{St}(x, R)$
R
$\partial\Omega_\mathrm{N}$
$\partial\mathrm{St}_\mathrm{B}$
$\partial\Omega_\mathrm{D}$
R = distance to closest pt on $\partial\Omega_\mathrm{D}$
$\partial\mathrm{St}_\mathrm{B}$
$\partial\mathrm{St}_\mathrm{N}$
$\mathrm{St}(x, R)$
R
x
$\partial\Omega_\mathrm{N}$
R = distance to closest silhouette pt on $\partial\Omega_\mathrm{N}$

Figure 5. WoSt defines a random walk on star-shaped regions $\mathrm{St}(x, R)$, formed by intersecting a ball $\mathrm{B}(x, R)$ with the domain Ω. The radius R is the minimum of the distances to the closest point on the Dirichlet boundary $\partial\Omega_\mathrm{D}$ (*left*) and the closest silhouette point on the Neumann boundary $\partial\Omega_\mathrm{N}$ (*right*). We use $\partial\mathrm{St}_\mathrm{N}$ and $\partial\mathrm{St}_\mathrm{B}$ to denote visible parts of $\partial\Omega_\mathrm{N}$ and $\partial\mathrm{B}$ (inside Ω) from x (*resp.*). For a Robin boundary $\partial\Omega_\mathrm{R}$, we use $\partial\mathrm{St}_\mathrm{R}$ in place of $\partial\mathrm{St}_\mathrm{N}$.



▢ epsilon shell ε
🟩 visible from x_k (sampled)
■ hidden from x_k (not sampled)
x_k
v
x_{k+1}
$\bar{x}_{k+1}$
ε
$\partial\Omega_\mathrm{D}^\varepsilon$
$\partial\Omega_\mathrm{D}$
x_{k+1}
v
x_k
ε
$\partial\Omega_\mathrm{N}^\varepsilon$
$\partial\Omega_\mathrm{N}$

Figure 6. WoSt uses an ε-shell to terminate walks on a Dirichlet boundary and to prevent walks from stopping on a concave Neumann boundary. *Left:* Inside $\partial\Omega_\mathrm{D}^\varepsilon$, WoSt uses the known Dirichlet data g from the closest projected point on $\partial\Omega_\mathrm{D}$ before terminating a walk. *Right:* Inside $\partial\Omega_\mathrm{N}^\varepsilon$, WoSt uses balls with minimum radius ε if the distance to the closest silhouette point is zero—only parts of $\partial\Omega_\mathrm{N}$ directly visible to x_k are sampled inside any ball $\mathrm{B}(x_k, \varepsilon)$, which implicitly assumes that the BVP solution is zero elsewhere.

3.2 Walk on Stars for Dirichlet-Neumann Conditions

We describe the original WoSt algorithm by Sawhney et al. [2023] for BVPs with mixed Dirichlet and Neumann conditions, *i.e.*, Equation 1 with $\mu = 0$. The pseudocode in Algorithm 1, annotated with comments in gray, corresponds to this version of WoSt. We extend WoSt to Robin boundary conditions in Section 4, and label the resulting modifications in Algorithm 1 in green.

3.2.1 Star-Shaped Regions. While the original walk on spheres algorithm [Muller 1956] for pure Dirichlet problems performs random walks using balls wholly contained in Ω (*top left*, Figure 2), WoSt instead performs walks that use *star-shaped regions* relative to a point $x \in \Omega$, *i.e.*, regions whose boundary is visible from x. WoSt forms such regions $\mathrm{St}(x, R)$ by intersecting the domain Ω with a ball $\mathrm{B}(x, R)$ centered at x; the ball's radius R equals the minimum of the distances from x to the closest point on $\partial\Omega_\mathrm{D}$, and the *closest silhouette point* on $\partial\Omega_\mathrm{N}$ (*right*, Figure 5). Thus, B can contain parts of $\partial\Omega_\mathrm{N}$ visible from x, but not $\partial\Omega_\mathrm{D}$. We denote by $\partial\mathrm{St}_\mathrm{N} := \partial\Omega_\mathrm{N} \cap \partial\mathrm{St}$ the Neumann part of the boundary of St, which has prescribed derivative values $\partial u/\partial n = h$ from the boundary condition in Equation 1 (with $\mu = 0$). We use $\partial\mathrm{St}_\mathrm{B} := \partial\mathrm{B} \cap \partial\mathrm{St}$ for the spherical part of $\partial\mathrm{St}$.

With A = St and C = B, the integral in Equation 2 becomes

$$\alpha(x)\, u(x) = \int_{\partial\mathrm{St}(x,R)} P^\mathrm{B}(x, z)\, u(z)\, \mathrm{d}z - \int_{\partial\mathrm{St}_\mathrm{N}(x,R)} G^\mathrm{B}(x, z)\, h(z)\, \mathrm{d}z + \int_{\mathrm{St}(x,R)} G^\mathrm{B}(x, y)\, f(y)\, \mathrm{d}y. \quad (3)$$

The critical simplification compared to Equation 2 is that $u(z)$ is the only unknown in this equation: the derivative $\partial u/\partial n_z(z)$ is known on $\partial\mathrm{St}_\mathrm{N}$, and is not needed on $\partial\mathrm{St}_\mathrm{B}$ where $G^\mathrm{B}(x, z) = 0$. From Equation 3, Sawhney et al. [2023] proceed to use recursive Monte Carlo estimation to develop the WoSt algorithm for BVPs with Dirichlet and Neumann conditions (Section 3.2.2). In Section 4, we likewise first derive an analogue of Equation 3 for Robin problems that also uses star-shaped regions, but we change the radius R to account for Robin conditions. We then use recursive Monte Carlo estimation on the resulting BIE to extend WoSt to BVPs with Robin conditions.

3.2.2 Monte Carlo Estimator. The Monte Carlo method approximates an integral $I := \int_A \phi(x)\, \mathrm{d}x$ using the sum

$$\widehat{I} := \frac{1}{N} \sum_{n=1}^{N} \frac{\phi(x^n)}{p^\mathrm{A}(x^n)}, \qquad x^n \sim p^\mathrm{A}, \quad (4)$$

where x^n are independent random samples from a probability density p^A that is non-zero on the support of ϕ. Using *single-sample* Monte Carlo (*i.e.*, $N = 1$) results in the *WoSt estimator* for any $k \geq 0$.

Walk on Stars With Dirichlet-Neumann conditions

$$\widehat{u}(x_k) = \frac{P^\mathrm{B}(x_k, x_{k+1})\, \widehat{u}(x_{k+1})}{\alpha(x_k)\, p^{\partial\mathrm{St}(x_k,R)}(x_{k+1})} - \frac{G^\mathrm{B}(x_k, z_{k+1})\, h(z_{k+1})}{\alpha(x_k)\, p^{\partial\mathrm{St}_\mathrm{N}(x_k,R)}(z_{k+1})} + \frac{G^\mathrm{B}(x_k, y_{k+1})\, f(y_{k+1})}{\alpha(x_k)\, p^{\mathrm{St}(x_k,R)}(y_{k+1})}. \quad (5)$$

This estimator is *recursive* as $\widehat{u}$ appears on both sides of Equation 5. Recursion leads to a *random walk* from one star-shaped region to another—hence the name *walk on stars.* At each step k, WoSt samples points $x_{k+1} \in \partial\mathrm{St}$, $z_{k+1} \in \partial\mathrm{St}_\mathrm{N}$ and $y_{k+1} \in \mathrm{St}$ from densities $p^{\partial\mathrm{St}}$, $p^{\partial\mathrm{St}_\mathrm{N}}$ and p^{St} (*resp.*). We discuss sampling x_{k+1} below, and refer to Sawhney et al. [2023, Sections 4.5 & 4.6] for details on sampling z_{k+1} and y_{k+1} to accumulate (non-recursive) contributions from the known Neumann data h and source f. For problems with a Dirichlet boundary $\partial\Omega_\mathrm{D}$, WoSt terminates a walk when it enters the *ε-shell* $\partial\Omega_\mathrm{D}^\varepsilon := \{x \in \Omega : \min_{y \in \partial\Omega_\mathrm{D}} \|x - y\| \leq \varepsilon\}$, and uses the Dirichlet data g at the closest point $\bar{x}_k \in \partial\Omega_\mathrm{D}$ to set $\widehat{u}(x_k) := g(\bar{x}_k)$ (*left*, Figure 6). For pure Neumann problems, WoSt uses *Tikhonov regularization* [Sawhney et al. 2023, Section 3.4.3] to terminate walks at the expense of some bias. No such regularization is needed for pure Robin problems as walks can terminate on $\partial\Omega_\mathrm{R}$ (Section 4.3).

3.2.3 Sampling Star-Shaped Regions. WoSt selects the next walk location x_{k+1} by *importance sampling* the Poisson kernel $P^\mathrm{B}(x_k, x_{k+1})$ in the recursive (first) term of Equation 5. Because the Poisson kernel equals the *signed solid angle* subtended by $\partial\mathrm{St}$ at x_k [Sawhney et al. 2023, Equation 25], WoSt can use *direction sampling* to select x_{k+1} easily, *i.e.*, similar to Monte Carlo ray tracing, we cast a ray from $x_k \in \Omega$ in a direction uniformly sampled on the unit sphere, and set x_{k+1} equal to the first intersection with $\partial\mathrm{St}$. If x_k lies on $\partial\Omega_\mathrm{N}$, WoSt instead uses *hemispherical* direction sampling to ensure $x_{k+1} \in \Omega$ (*right*, Figure 6), which also prevents the estimate $\widehat{u}(x_{k+1})$ from picking up a multiplicative factor of 2 each time a walk reaches $\partial\Omega_\mathrm{N}$. This is because the 1/2 factor from sampling a hemisphere (instead of a sphere) cancels $\alpha(x_k) = 1/2$ in the denominator of Equation 5. Together, these sampling decisions guarantee that walk throughput (*i.e.*, an accrued multiplicative factor for $\widehat{u}(x_{k+1})$ over k steps) is always 1, as $p^{\partial\mathrm{St}} = P^\mathrm{B}(x_k, x_{k+1})/\alpha(x_k)$ at each step k.

3.2.4 Epsilon Clamp. WoSt limits the radius R of $\mathrm{St}(x_k, R)$ to be greater than a user-defined ε. Doing so ensures that walks do not stall near concave parts of a Neumann boundary, where the distance to the closest silhouette point on $\partial\Omega_\mathrm{N}$ shrinks to zero [Sawhney et al. 2023, Figure 9]. Akin to the ε-shell $\partial\Omega_\mathrm{D}^\varepsilon$, this scheme introduces a performance-bias tradeoff: bias stems from $\mathrm{St}(x_k, \varepsilon)$ not necessarily being star-shaped, which means that the solution u is effectively zero at points $x_{k+1} \in \partial\mathrm{St}$ not visible from x_k (*right*, Figure 6). For Robin problems, we will also apply an ε-clamp to each St, and use direction sampling to determine the next walk location x_{k+1}.

4 WALK ON STARS FOR ROBIN CONDITIONS

To add support for BVPs with Robin boundary conditions to WoSt, our main modification is to change how we select the radius R of the balls $\mathrm{B}(\cdot, R)$ we use to form star-shaped regions St, leaving the rest of the algorithm largely unchanged. As we show in Figure 3, the Robin coefficient μ 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 Figure 7, we show that R achieves its maximal value R_0 on a fixed boundary when $\mu = 0$ (*i.e.*, $\partial\Omega_\mathrm{R} \equiv \partial\Omega_\mathrm{N}$), with $\mathrm{B}(\cdot, R)$ 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 $\mathrm{B}(\cdot, R)$ 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 $\mathrm{B}(\cdot, R)$ expands.

In the rest of this section, we formalize this intuition for the ball size in three steps: in Section 4.1, we introduce a reflectance function ρ_μ, which adds support for Robin conditions to the BIE we use to derive our subsequent WoSt estimator. In Section 4.2, 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. Finally in Section 4.3, we show how ρ_μ allows terminating random walks early through Russian roulette [Pharr et al. 2016, Section 13.7] to improve efficiency. We highlight these changes in Algorithm 1 in green.

4.1 Modified BIE and MC Estimator

To derive a boundary integral that accounts for Robin conditions, we follow largely the same derivation as that of Equation 3 [Sawhney et al. 2023, Section 4.2]. The main difference is that we use the *Brakhage-Werner trick* from potential theory [Nédélec 2001] to

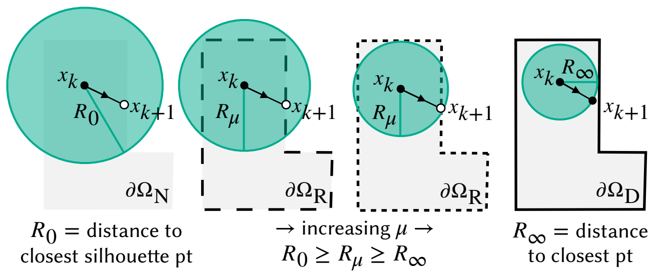

x_k
R_0
x_{k+1}
$\partial\Omega_\mathrm{N}$
x_k
R_μ
x_{k+1}
$\partial\Omega_\mathrm{R}$
x_k
R_μ
x_{k+1}
$\partial\Omega_\mathrm{R}$
x_k
R_∞
x_{k+1}
$\partial\Omega_\mathrm{D}$
R_0 = distance to closest silhouette pt
→ increasing μ →
$R_0 \geq R_\mu \geq R_\infty$
R_∞ = distance to closest pt

Figure 7. By increasing μ, the radius of our star-shaped region for a Robin boundary reduces continuously from the distance to the closest silhouette point (*left:* Neumann case with $\mu = 0$) to the distance to the closest point on the boundary (*right:* Dirichlet case with $\mu = \infty$).

substitute $\partial u/\partial n = \mu \cdot u + h$ (from the Robin condition in Equation 1) on $\partial\mathrm{St}_\mathrm{R} := \partial\Omega_\mathrm{R} \cap \partial\mathrm{St}$. Rearranging terms then yields:

$$\begin{aligned} \alpha(x)\, u(x) = &\int_{\partial\mathrm{St}(x,R)} \rho_\mu(x,z)\, P^\mathrm{B}(x,z)\, u(z)\, \mathrm{d}z \\ &- \int_{\partial\mathrm{St}_\mathrm{R}(x,R)} G^\mathrm{B}(x,z)\, h(z)\, \mathrm{d}z \\ &+ \int_{\mathrm{St}(x,R)} G^\mathrm{B}(x,y)\, f(y)\, \mathrm{d}y, \end{aligned} \tag{6}$$

where we define the spatially-varying function ρ_μ as:

$$\rho_\mu(x,z) := \begin{cases} 1 - \mu(z)\, \frac{G^\mathrm{B}(x,z)}{P^\mathrm{B}(x,z)}, & \text{on } \partial\mathrm{St}_\mathrm{R}, \\ 1, & \text{on } \partial\mathrm{St}_\mathrm{B}. \end{cases} \tag{7}$$

Equation 6 is nearly identical to Equation 3 for BVPs with Neumann conditions, and even reduces to it when $\mu = 0$. Importantly, as before, u is the only (recursively-defined) unknown in Equation 6. Similar to Section 3.2.2, we can therefore use single-sample Monte Carlo to derive a recursive *WoSt estimator with Robin conditions.*

Walk on Stars With Robin conditions

$$\begin{aligned} \widehat{u}(x_k) = &\frac{\rho_\mu(x_k, x_{k+1})\; P^\mathrm{B}(x_k, x_{k+1})\, \widehat{u}(x_{k+1})}{\alpha(x_k)\; p^{\partial\mathrm{St}(x_k,R)}(x_{k+1})} \\ &- \frac{G^\mathrm{B}(x_k, z_{k+1})\, h(z_{k+1})}{\alpha(x_k)\; p^{\partial\mathrm{St}_\mathrm{R}(x_k,R)}(z_{k+1})} + \frac{G^\mathrm{B}(x_k, y_{k+1})\, f(y_{k+1})}{\alpha(x_k)\; p^{\mathrm{St}(x_k,R)}(y_{k+1})}. \end{aligned} \tag{8}$$

Given how similar the integrands are in Equations 3 and 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 $p^{\partial\mathrm{St}}, p^{\partial\mathrm{St}_\mathrm{R}}, p^{\mathrm{St}}$ (*resp.*) as in Sections 3.2.2–3.2.3. Apart from the introduction of ρ_μ, the WoSt estimator for Robin conditions is unchanged from the estimator for Dirichlet-Neumann conditions. Next, we show how to select the radius R for each star-shaped region $\mathrm{St}(x_k, R)$, and how to terminate random walks on $\partial\Omega_\mathrm{R}$—these are the only two places our estimator deviates from Equation 5.

ALGORITHM 1: WalkOnStars($x,\ n_x,\ \varepsilon$)
Note: Code annotated with comments in green indicates our modifications to the WalkOnStars algorithm by Sawhney et al. [2023].

```
Input: Starting position $x \in \Omega$ of random walk, normal $n_x$ at $x$ (undefined if $x \notin \partial\Omega_R$), $\varepsilon$-shell parameter.
Output: Single-sample Monte Carlo estimate $\widehat{u}(x)$ for Equation 1.
1:  $d,\ \overline{x} \leftarrow$ DistanceAbsorbingBoundary($x$)    ▷Compute distance to absorbing boundary $\partial\Omega_D$ with Dirichlet conditions ($\infty$ if $\partial\Omega_D = \emptyset$)
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:  $R \leftarrow$ StarRadiusReflectingBoundary($x,\ d$)    ▷Compute radius of star region $\mathrm{St}(x,R)$ containing reflecting boundary $\partial\Omega_R$, such that $R \leq d$ (Alg. 2)
4:  $R \leftarrow \max(\varepsilon,\ R)$    ▷Also ensure $R \geq \varepsilon$
5:  $v \leftarrow$ SampleUnitSphere()    ▷Sample a direction $v$ uniformly on the unit sphere
6:  if $x \in \partial\Omega_R$ and $n_x \cdot v > 0$ then $v \leftarrow -v$    ▷If $x$ lies on $\partial\Omega_R$, ensure $v$ is sampled on hemisphere with axis $-n_x$
7:  hit, $p,\ n_p \leftarrow$ IntersectReflectingBoundary($x,\ v,\ R$)    ▷Intersect $\partial\mathrm{St}_R$ (boundary inside star region) with ray $x + Rv$, and get first hit
8:  if not hit then $p \leftarrow x\ +\ R\,v$    ▷If there is no hit with $\partial\mathrm{St}_R$, intersect $\partial\mathrm{St}_B$ (spherical portion of star region) instead
9:  $\widehat{I}_h \leftarrow$ ReflectingBoundaryEstimate($x,\ R$)    ▷Estimate contribution from boundary term $h$ on $\partial\mathrm{St}_R$ [Sawhney et al. 2023, Alg. 1, lines 18-22]
10: $\widehat{I}_f \leftarrow$ SourceEstimate($x,\ p,\ v,\ R$)    ▷Estimate contribution from source term $f$ in St [Sawhney et al. 2023, Alg. 1, lines 24-26]
11: $\rho_\mu \leftarrow$ Clamp($1 - \mu(p) G^{\mathrm{B}(x,R)}(x,\ p) / P^{\mathrm{B}(x,R)}(x,\ p),\ 0,\ 1$)    ▷Compute reflectance (Eq. 7) and clamp it to $[0,1]$
12: if $\rho_\mu <$ SampleUniform(0, 1) then return $-\widehat{I}_h\ +\ \widehat{I}_f$    ▷Attempt to terminate walk using Russian roulette
13: return WalkOnStars($p,\ n_p,\ \varepsilon$) $-\ \widehat{I}_h\ +\ \widehat{I}_f$    ▷Repeat procedure from updated walk position ($n_p$ is undefined if $x \notin \partial\Omega_R$)
```

4.2 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 $\partial\Omega_R$ from x, *i.e.*, the radius R_0 we use for Neumann problems (*middle*, Figure 8). To understand why this is a bad choice for R, we must consider the values $\rho_\mu(x,z)$ (Equation 7) assumes on different parts of $\partial\mathrm{St}(x,R)$. In particular, irrespective of the value of μ, $\rho_\mu(x,z) = 1$ at points $z \in \partial\mathrm{St}_B$, as $G^{\mathrm{B}}(x,z) = 0$. When $\mu = 0$, $\rho_\mu(x,z)$ likewise simplifies to 1 at points $z \in \partial\mathrm{St}_R$, 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 leads to extremely high variance in the *recursive* estimator in Equation 8 for two reasons: (1) The solution estimate $\widehat{u}$ accumulates a *throughput*[1] $\prod_k \rho_\mu(x_{k-1}, x_k)$ that becomes unbounded in magnitude as walk length k increases. (2) The function ρ_μ, and thus the throughput, can change sign along a walk, resulting in unstable estimates due to cancellations [Kalos and Whitlock 2009, Chapter 4]. In Appendix A, we provide an operator-theoretic analysis of boundary integral equations to more formally explain the issues with a naïve estimation of Equation 6 and our solution to it, which we discuss below.

4.2.1 Shrinking the radius. To ensure that 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 $\mathrm{B}(x,R)$ [Sawhney et al. 2023, Equations 25 & 26] into Equation 7. For any point $z \in \partial\mathrm{St}_R$, we have

$$\rho_\mu(x,z) = 1 - \frac{\mu(z)\ r}{\cos\theta}\left(1 - \frac{r}{R}\right), \tag{9}$$

where $r = \|z - x\|$, $\cos\theta = {}^{(n_z \cdot (z-x))}/_{r}$. The terms $1 - {}^{r}/_{R}$ and $\cos\theta$ are both positive, as $r \leq R$ 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

$$\frac{\mu(z)\ r}{\cos\theta}\left(1 - \frac{r}{R}\right) \leq 1. \tag{10}$$

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

$$R \leq \frac{r}{1 - \frac{\cos\theta}{\mu(z)\ r}} \quad \text{when } r > \frac{\cos\theta}{\mu(z)}, \tag{11}$$

which must hold at *all* points $z \in \partial\mathrm{St}_R$ (*right*, Figure 8). 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 Equation 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 ∞ (*resp.*). With this bound for the star-shaped region radius R, we term ρ_μ the *reflectance* for the Robin boundary inside $\mathrm{St}(x,R)$, as it encodes

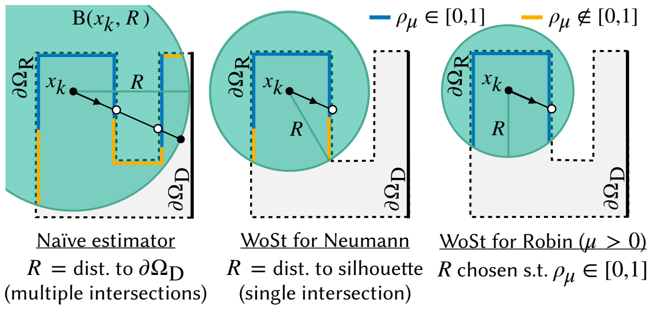


Figure 8. We use reflectance values ρ_μ on the Robin boundary $\partial\Omega_R$ to determine the radius R of a star-shaped region $\mathrm{St}(x_k, R)$. *Left:* For a ball $\mathrm{B}(x_k, R)$ where R is the distance to the Dirichlet boundary $\partial\Omega_D$, the BVP solution must be estimated at all ray intersections sampled proportionally to signed solid angle. *Middle:* WoSt with Neumann conditions avoids multiple intersections by 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 $\rho_\mu \in [0,1]$.

[1]We use this term in analogy with the *throughput* of a light path in Monte Carlo rendering [Pharr et al. 2016]. In our setting, the throughput of a walk equals the probability with which Brownian motion is reflected off (and not absorbed on) $\partial\Omega_R$.

the probability with which a random walk is reflected off $\partial\mathrm{St}_\mathrm{R}$. In Section 5.2, we describe how to compute R efficiently on triangle meshes, with only small modifications to how $d_\mathrm{silhouette}$ is computed for $\partial\Omega_\mathrm{N}$ [Sawhney et al. 2023, Section 5.1]. Appendix B gives corresponding expressions for 2D domains.

4.2.2 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 $\partial\Omega_\mathrm{R}$. As in Section 3.2.4, we therefore clamp $R \geq \varepsilon$ to prevent walks from stalling near $\partial\Omega_\mathrm{R}$. As $\mathrm{St}(x_k, \varepsilon)$ may no longer be star-shaped, we also clamp ρ_μ to $[0, 1]$ to ensure that throughput remains bounded (*line 11*, Algorithm 1). 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 Figure 11, we use different ε values to study the impact clamping has on bias and performance.

4.3 Using Russian Roulette to Terminate Walks

Using direction sampling (Section 3.2.3) to select the next walk location x_{k+1} perfectly importance samples (and hence cancels out) the Poisson kernel $P^\mathrm{B}(x_k, x_{k+1})$ in Equation 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* [Pharr et al. 2016, Section 13.7]: 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 surviving walks (*line 12*, Algorithm 1). 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 $\partial\Omega_\mathrm{D}^\varepsilon$ while their throughput continues to shrink. This often leads to large efficiency gains, as we show in Figure 3. We note that Russian roulette is not possible with pure Neumann conditions, where reflectance always equals 1. In this case, a walk must continue until it reaches $\partial\Omega_\mathrm{D}^\varepsilon$. Otherwise it never terminates when $\partial\Omega_\mathrm{D} = \emptyset$, which necessitates Sawhney et al. [2023, Section 3.4.3] to use Tikhonov regularization.

5 IMPLEMENTATION ON TRIANGLE MESHES

We require essentially the same geometric queries and acceleration structures as Sawhney et al. [2023] to implement Algorithm 1 on triangle meshes. The queries we need are:

(1) Closest point on $\partial\Omega_\mathrm{D}$ (*line 1*)
(2) Radius of star-shaped region for $\partial\Omega_\mathrm{R}$ (*line 3*)
(3) Ray intersection against $\partial\Omega_\mathrm{R}$ (*line 7*)
(4) Point sampling of known Robin data h on $\partial\Omega_\mathrm{R}$ (*line 9*)

Queries 1, 3, and 4 remain unchanged from Sawhney et al. [2023], and are supported by the `FCPW` library [Sawhney 2021]: ray intersections and closest point queries (CPQs) are standard in computer graphics and can be accelerated using a BVH; the point sampling query in Sawhney et al. [2023, Section 5.2] was designed to sample known Neumann data h on a reflecting boundary, but applies out-of-the-box to Robin data—it too is implemented using a BVH.

Here we discuss how to implement the radius query for $\partial\Omega_\mathrm{R}$, which functions as a closest silhouette point query (CSPQ) [Sawhney et al. 2023, Section 5.1] when $\mu = 0$, and as a CPQ when $\mu = \infty$. For intermediate coefficient values, it has additional aspects unique to Robin conditions, which we annotate in green in Algorithms 2, 3 and 4. Like Sawhney et al. [2023], we use an SNCH [Johnson and Cohen 2001] for acceleration, which adds information about boundary normals and Robin coefficients to a BVH (Figure 9).

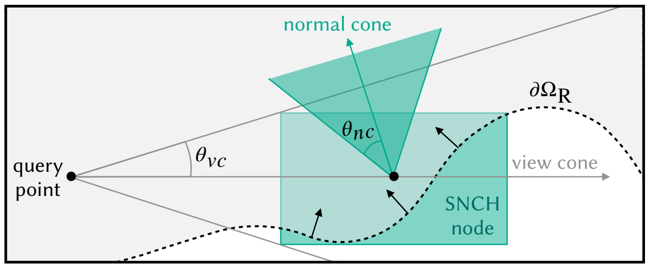


Figure 9. Like Sawhney et al. [2023], we employ a spatialized normal cone hierarchy to accelerate our star-shaped region query for a Robin boundary $\partial\Omega_\mathrm{R}$. We use the spatial and angular bounds for the geometry inside an SNCH node (via each node's bounding box and normal cone) to compute a conservative radius for the star-shaped region, which determines whether a node can be skipped as we traverse the hierarchy.

In practice, we use a BVH to find closest points on $\partial\Omega_\mathrm{D}$, and an SNCH for all other queries on $\partial\Omega_\mathrm{R}$ (though only the radius query accesses information about normals and coefficients). We implement our method on triangle meshes in the open-source `Zombie` library [Sawhney and Miller 2023], but it works with any boundary representation that supports Queries 1–4. The boundary can be open or closed, and may contain self-intersections, cracks or holes.

5.1 Computing Radius Bounds for Triangles

The naïve approach for computing the radius of a star-shaped region on $\partial\Omega_\mathrm{R}$ is to first perform a CSPQ on the mesh relative to a query point x, and then: loop over all triangles within the radius returned by the CSPQ, estimate the upper-bound on the radius for each triangle t (Equation 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 instead compute a tight bound using the maximum coefficient value[2] $\mu^\mathrm{max} := \max(\mu(z))$ for 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 Equation 11, we have:

$$R \leq \frac{\mu^\mathrm{max}\ h^2}{\mu^\mathrm{max}\ h\ \cos\theta - \cos^3\theta} \quad \text{when}\ \cos\theta \leq \sqrt{\mu^\mathrm{max}\ h}. \tag{12}$$

We now minimize this equation by taking its derivative with respect to $\cos\theta$, and setting it to zero. This gives us an analytical expression $\sqrt{\mu^\mathrm{max} h/3}$ for the cosine. We clamp this expression between the minimum and maximum cosine values achieved at the closest and farthest (*resp.*) points on t, and plug it back into Equation 12 to compute the radius bound for t. Algorithm 3 provides pseudocode.

5.2 Accelerating Star-Shaped 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

[2]Using μ^max to compute the radius bound for a triangle does not alter the original problem description in any way—our estimator still treats $\mu(z)$ as spatially varying over the triangle when computing reflectance in *line 11*, Algorithm 1.

parts of $\partial\Omega_R$ where the geometry is entirely front- or back-facing relative to the query point x (akin to a CSPQ). We therefore construct a hierarchical acceleration structure that stores the following information about the triangles in each node:

(1) An axis-aligned bounding box for spatial extents.
(2) A *normal cone* for angular extents.
(3) Minimum and maximum Robin coefficients.

We compute the cone axis by averaging all triangle normals in a node, and set the cone half angle equal to the maximum deviation of any triangle normal from the axis. We use the surface area heuristic [Wald 2007] to construct this SNCH; the normal cones and Robin coefficients for each node are assembled during construction.

Query Traversal. Similar to Section 5.1, we compute *conservative* radius bounds for the nodes we visit during traversal, using the spatial, angular, and coefficient information available in a node (*lines 7 & 8*, Algorithm 4). We also build a *view cone* rooted at x to compute the bounds, via Equation 11; the cone axis points towards the center of a node, and its half angle bounds the node tightly (Figure 9). We then decide whether to skip a node with only front- or back-facing triangles, by checking if our minimum bound for the node is larger than the current estimate of the radius for the star-shaped region (*line 1*, Algorithm 2). We also use our maximum bound for the node to shrink the radius estimate (*lines 7 & 9*, Algorithm 2). If instead the node contains a geometric silhouette (*line 4*, Algorithm 4), we must traverse the node just as with a CSPQ, as Equation 11 only applies to star-shaped regions that can be no larger in size than the distance returned by a CSPQ.

6 VARIANCE REDUCTION

We adapt two variance reduction techniques previously developed for WoSt [Miller et al. 2023; Qi et al. 2022] to Robin problems to further improve estimation quality (Figure 10). Adaptation is relatively straightforward, as our method only makes minor modifications to the original WoSt algorithm. We assume for simplicity that the domain boundary only has Robin conditions, and refer to the original papers for details regarding Dirichlet and Neumann conditions.

6.1 Boundary Value Caching

WoSt does not exploit the spatial smoothness in the solution to an elliptic PDE, as it estimates the solution independently at every point in Ω. To reduce redundant computation and suppress noise, *boundary value caching (BVC)* [Miller et al. 2023] instead first uses WoSt to estimate solution values u and derivatives $\partial u/\partial n$ at random points on $\partial\Omega$. BVC then uses these cached boundary values to directly evaluate the BIE at interior points via Monte Carlo, setting $\mathrm{A} = \Omega$ and $\mathrm{C} = \mathbb{R}^N$ in Equation 2. BVC generates smoother results than WoSt for Dirichlet and Neumann problems [Miller et al. 2023, Figures 1, 5 & 7], and facilitates cheap and output-sensitive evaluation of the PDE solution in Ω.

For BVPs with Robin conditions, we do not need to estimate both u and $\partial u/\partial n$ on $\partial\Omega_R$ to use BVC. Instead, we can substitute

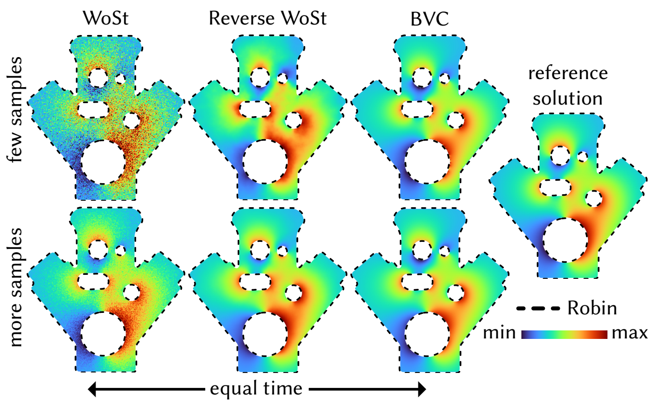


Figure 10. We extend bidirectional formulations for WoS [Qi et al. 2022] (*middle column*) and boundary value caching for WoSt [Miller et al. 2023] (*right column*) to BVPs with Robin conditions, and achieve smoother results with less noise compared to the WoSt pointwise estimator (*left column*).

$\partial u/\partial n = \mu u + h$ into the BIE used by Miller et al. [2023, Equation 2]:

$$\alpha(x)\,u(x) = \int_{\partial\Omega_R} \left(P^{\mathbb{R}^N}(x,z) - \mu(z)G^{\mathbb{R}^N}(x,z)\right) u(z)\,\mathrm{d}z - \int_{\partial\Omega_R} G^{\mathbb{R}^N}(x,z)h(z)\,\mathrm{d}z + \int_{\Omega} G^{\mathbb{R}^N}(x,y)f(y)\,\mathrm{d}y, \quad (13)$$

As the only unknown on the right-hand side is the solution u, we use our WoSt algorithm (Algorithm 1) to estimate its values at random points $z \in \partial\Omega_R$. We then evaluate this BIE at interior points $x \in \Omega$ with Monte Carlo using the cached values for $u(z)$. Miller et al. [2023, Appendix A] provide explicit expressions for $G^{\mathbb{R}^N}$ and $P^{\mathbb{R}^N}$.

We note that as an alternative to Equation 13, we could also make the substitution $u = (\partial u/\partial n - h)/\mu$ when $\mu > 0$, and instead estimate (and cache) values of $\partial u/\partial n$ on $\partial\Omega_R$: Sawhney and Crane [2020, Section 3.1] and Miller et al. [2023, Section 3] discuss how to compute normal derivatives with WoSt, and their estimators work with Robin conditions as well. Empirically, we observe that estimating $\partial u/\partial n$ on $\partial\Omega_R$ typically works better when μ is large. This behavior is because BVC does not importance sample the $P - \mu G$ term in Equation 13 when generating cache samples on $\partial\Omega_R$, and thus a large μ amplifies the noise in estimated values for u on $\partial\Omega_R$.

6.2 Reverse Random Walks

In place of estimating u or $\partial u/\partial n$ on the boundary as with BVC, Qi et al. [2022] 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. As with bidirectional algorithms for light transport [Lafortune and Willems 1993; Veach and Guibas 1995], these walks can be more efficient than "forward" walks to the boundary (such as those in Sections 3–4), as a single reverse walk contributes to the solution estimate at multiple points in Ω.

To derive a similar reverse walk algorithm for Robin (and Neumann) problems, we substitute $\partial u/\partial n = \mu u + h$ into Equation 2 as

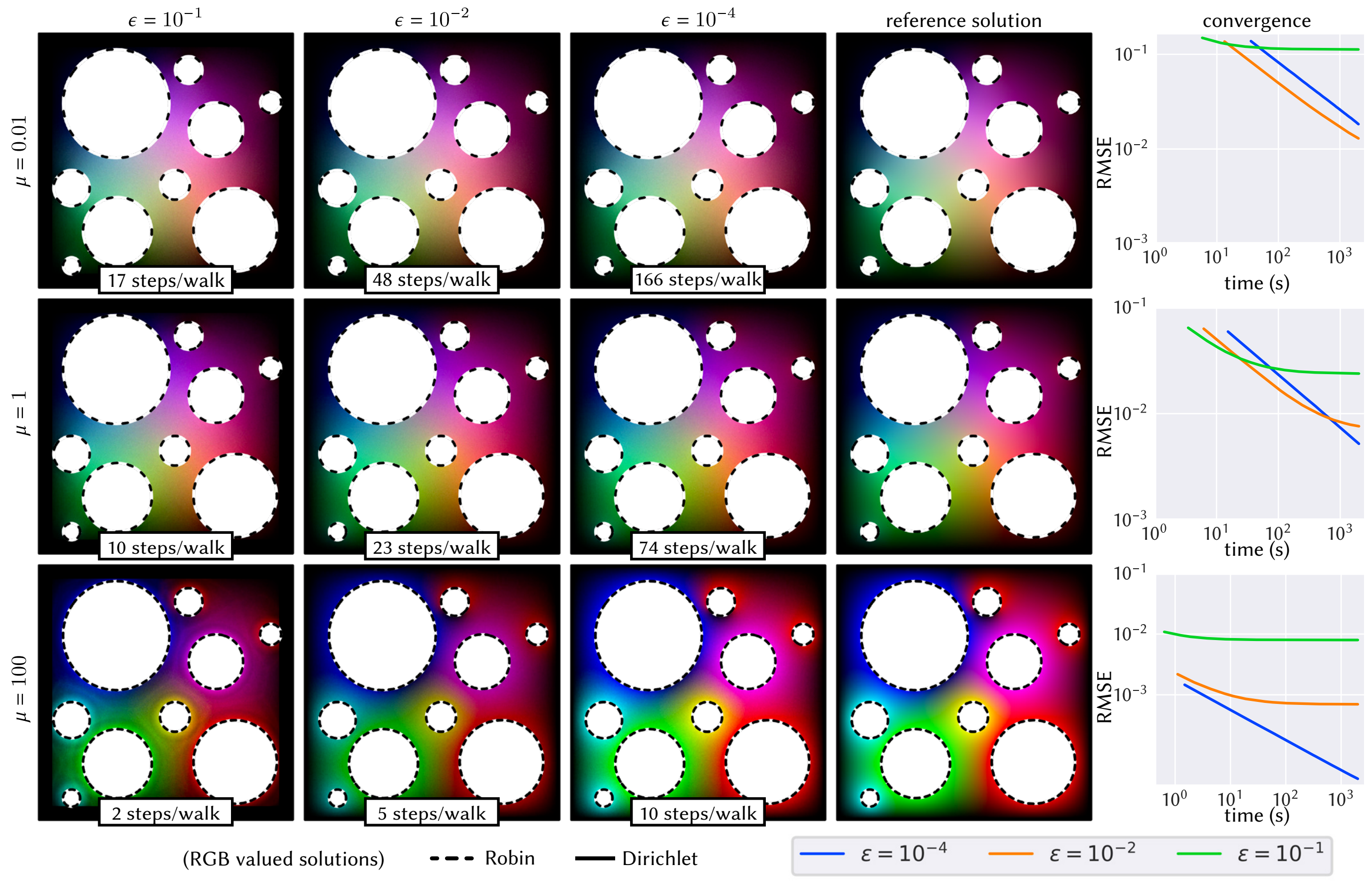


Figure 11. WoSt uses a single ε-shell parameter to control the tradeoff between bias in a solution estimate and the number of steps in a walk—in general, this parameter requires little-to-no hand-tuning as bias drops predictably with decreasing ε values. *Top two rows:* For more reflecting Robin boundaries with smaller coefficients μ, bias manifests as a global darkening in the solution estimate for large ε, with runtime improvements typically outweighing the relative increase in bias. *Bottom row:* For more absorbing Robin boundaries with larger coefficients μ, a large ε-shell produces a Voronoi-like solution that extends prescribed boundary values further into the domain interior—a similar bias is observed with WoS for pure Dirichlet problems [Sawhney and Crane 2020, Fig. 14]. Bias quickly goes away as ε decreases, with only a small increase in the average number of steps per walk.

before, but use the sets A = Ω and C = Ω instead. This yields

$$\alpha(x)\, u(x) = -\int_{\partial\Omega_R} G^{\Omega}(x,z)\, h(z)\, dz + \int_{\Omega} G^{\Omega}(x,y)\, f(y)\, dy. \quad (14)$$

The Green's function G^{Ω} is the *fundamental solution* of the BVP

$$\begin{array}{rcll} \Delta G^{\Omega}(x,y) & = & \delta_x(y) & \text{on } \Omega, \\ \frac{\partial G^{\Omega}(x,y)}{\partial n_y} - \mu(y) G^{\Omega}(x,y) & = & 0 & \text{on } \partial\Omega_R. \end{array} \quad (15)$$

Compared to Equation 6 or 13, there are no unknown solution values u in the boundary integral in Equation 14, as the Robin conditions on G^{Ω} zero them out. However, to compute u anywhere in the domain, we must now estimate G^{Ω} at points $z \in \partial\Omega_R$ and $y \in \Omega$, which we do using WoSt—we refer to Appendix C for details. We use the resulting reverse walks to improve solution estimates at multiple points $x \in \Omega$, by splatting contributions from known Robin data h (and source f). As with BVC, reverse walks produce smoother results than WoSt, albeit with more correlation artifacts at low sample counts (*middle column*, Figure 10), and less control over evaluation points in Ω.

7 EVALUATION

In this section, we study various practical aspects of our Monte Carlo algorithm, such as the impact of stopping tolerances on bias and performance (Section 7.1), the reliability of solution estimates in highly non-convex domains (Section 7.2), and robustness and scalability with increasing geometric complexity (Section 7.3). We use polygonal meshes to represent the boundary ∂Ω, and use callback functions to encode f, g, h and μ in Equation 1. We prototype our geometric queries using a CPU-based SNCH (Section 5.2), which does not have a significant preprocessing cost even for large models (often within 2× of the time needed to build a BVH [Wald 2007])—in contrast, finite element generation can take minutes to hours and can be very memory intensive (see Figure 4). We use a 12-core i9-10920X Intel CPU for all experiments with our method except Figure 1, for which we use a 64-core 3rd generation Intel workstation. We use the finite element library *MFEM* [Anderson et al. 2021] to compute reference solutions for Figures 10, 11 and 12.

7.1 Stopping Tolerances and Convergence

Our algorithm uses a single parameter to control the thickness of the ε-shell $\partial\Omega^{\varepsilon}$, irrespective of the type of boundary condition on $\partial\Omega$. As we discuss in Sections 3.2 and 4.2, as ε increases, walks typically take larger steps on $\partial\Omega_{\mathrm{N}}$ and $\partial\Omega_{\mathrm{R}}$, and terminate faster on $\partial\Omega_{\mathrm{D}}$. Figure 11 examines the impact of this parameter on a Laplace equation with both more reflecting ($\mu < 1$) and more absorbing ($\mu > 1$) Robin conditions. In both cases, runtime improvements from using larger ε outweigh the relative increase in bias—similar results have been observed with Dirichlet and Neumann conditions [Sawhney and Crane 2020; Sawhney et al. 2023, Figures 14 & 13]. We use 0.001× the diagonal scene length as our default ε value.

WoSt exhibits the expected $O(1/\sqrt{N})$ rate of convergence with respect to the number of walks N, which suggests that any bias has little impact on accuracy. Moreover, we observe predictable convergence in both convex and concave domains (Figure 12), though results are typically noisier with more reflecting Robin conditions.

7.2 Comparison With 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 (*left*, Figure 8). To avoid a branching walk that increases exponentially in size, WoB instead uses just a single randomly selected intersection. This results in extremely high variance even in domains that are mostly convex (Figure 12), 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. Consequently, by accumulating such multiplicative factors, the walk contribution grows unbounded as walk length increases, resulting in unbounded variance as noted by Sabelfeld and Simonov [2016, Chapter 2]. Moreover, the Poisson kernel alternates sign between consecutive intersections in the WoB estimator [Sugimoto et al. 2023, Section 4.1.1], which further results in unstable estimates due to cancellation [Kalos and Whitlock 2009, Chapter 4]. Sugimoto et al. [2023, Section 4.1.2] therefore propose truncating walk length to reduce variance, but as we show in Figure 12 (*left column*), doing so introduces significant bias in non-convex domains. 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.

WoB has even more trouble with Robin problems (*third and fourth row*, Figure 12), as it has no mechanism to deal with the reflectance term $\rho_\mu = 1 - \mu\, G/P$ in its integral expression (*e.g.*, via importance sampling). As a result, walk throughout typically grows even faster with Robin conditions, especially as μ increases—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 ∂St. Therefore, we do not truncate walk length to a predefined value like WoB, and instead use Russian roulette to terminate walks *without any additional bias*.

Figure 12 shows equal-time comparisons between WoB and WoSt for a Laplace BVP, where we run the reference WoB implementation from Sugimoto et al. [2023] on an Nvidia RTX 3090 GPU, and our WoSt implementation on a 12-core CPU—giving WoB the benefit of faster compute hardware. WoSt demonstrates stable convergence with little bias for Dirichlet, Neumann, and Robin conditions. Though WoSt takes on average more steps per walk than WoB (Table 1), the relative mean squared errors show that WoB is extremely sensitive to the walk length choice, and requires enormous sample count to converge, even to a solution with large bias.

7.3 Thermal Analysis

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 [Cadence 2024] and residential architecture [Kamel and Kazemian 2023] to spacecrafts and robotics. NASA itself 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."* [NASA 1999]. Just as computer graphics has long enjoyed the ability to iterate on illumination for virtual environments (via Monte Carlo rendering), the solver we develop here can help engineers to achieve the same kind of rapid and quantitatively reliable feedback during the design process—rather than waiting on bottlenecks like mesh generation (Figure 4).

Figure 1 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 robot surface by solving a Laplace BVP with Robin boundary conditions. Robin boundary data is given by thermal radiation from the Sun, computed via ordinary ray tracing. As 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 boundary conditions provides the opportunity for far more accurate physical modeling than purely absorbing (Dirichlet) or purely reflecting (Neumann) conditions alone. More accurate simulation might be obtained by coupling our solver with one that models, *e.g.*, convective heat transfer [Bati et al. 2023], though the low density of the Martian atmosphere makes this term largely negligible [Von Arx and Delgado Jr 1991].

Finally, Figure 1, *top right* illustrates a *deferred shading* approach [Deering et al. 1988] that is quite natural in the Monte Carlo setting, but has not been considered in prior work on WoS methods. Rather than evaluate 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 (Figure 1, *(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 points that actually need to be inspected for engineering analysis—Figure 1, *bottom* shows a collection of closeup viewpoints solved in the same fashion. Moreover, as Monte Carlo accumulates a running sum, we can quickly visualize

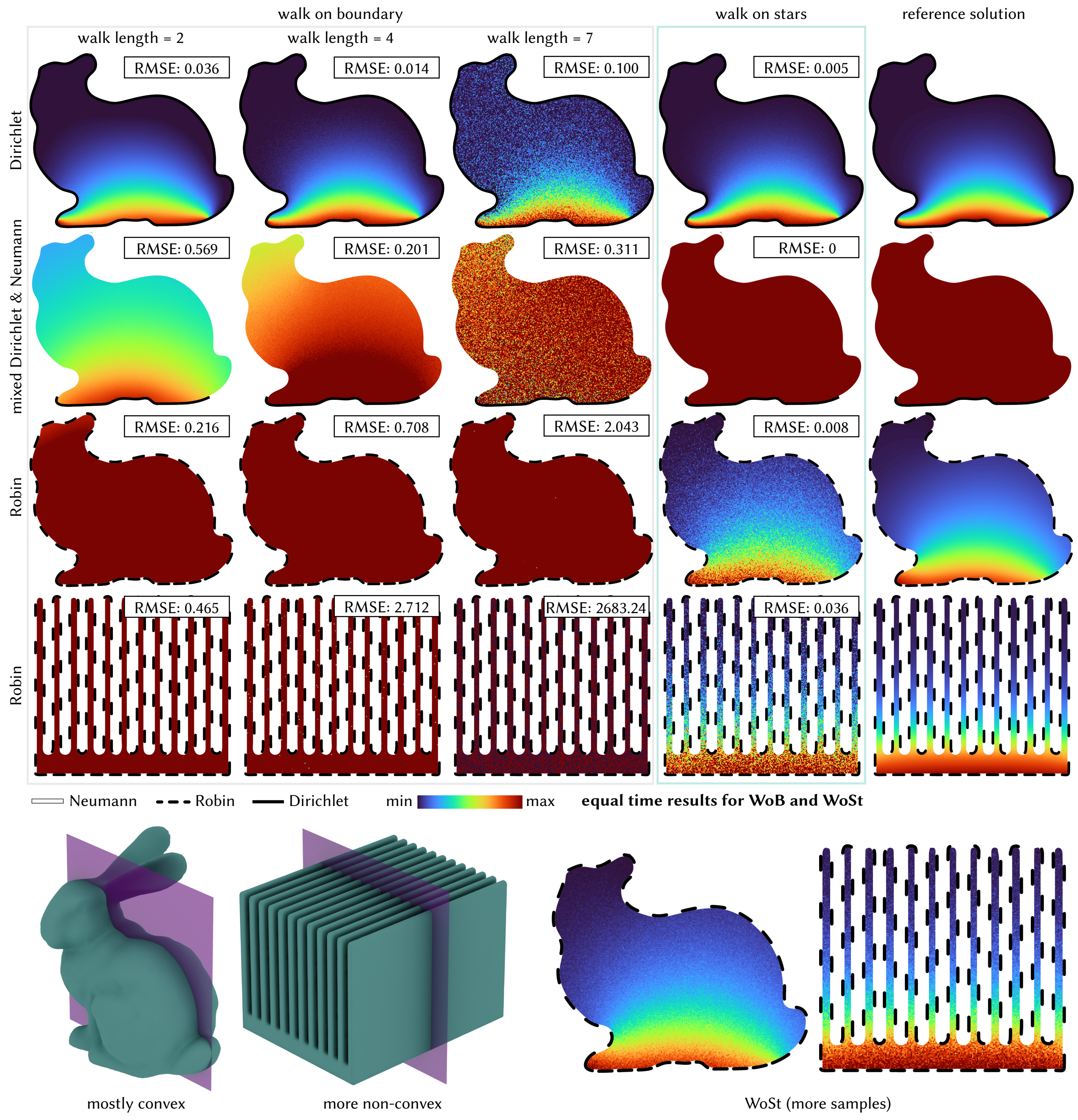


Figure 12. The WoB method suffers from an extreme bias-variance tradeoff, whereas WoSt demonstrates reliable Monte Carlo convergence with increasing sample count (*i.e.*, number of walks) for *any* combination of Dirichlet, Neumann and Robin boundary conditions. *First three rows:* Even in a mostly convex domain with simple boundary conditions, WoB has noticeable bias in its solution estimates 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. Furthermore, 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. By contrast, estimation with WoSt is equally stable in both convex and non-convex domains.

Table 1. Minimum and maximum estimated solution values, total number of walks, and average walk length for the WoB and WoSt results in Figure 12. Though WoSt generally requires longer walks than WoB, its solution estimates have significantly less error compared to WoB *at equal time* with fewer walks per point.

	estimator	WoB(2)	WoB(4)	WoB(7)	WoSt(ε=0.001)
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	$1.6\text{x}10^2$

a rough estimate of the solution that progressively improves over time (Figure 1, *bottom row of top right*). Across all four viewpoints we compute the solution at 793,000 points; for this model, our unoptimized CPU implementation takes 0.48 ms per walk, and computes a preview in about 1.5 min per viewpoint.

The complexity and significant nonconvexity of the rover geometry make this problem essentially intractable for the walk on boundary method [Sabelfeld and Simonov 2013; Sugimoto et al. 2023]. Likewise, 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 (Figure 4). 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.

8 CONCLUSION AND FUTURE WORK

We consolidate the treatment of first-order linear boundary conditions with WoSt, which allows us to model boundary value problems with much greater physical realism, and often better efficiency. The key strength of our method is that it functions reliably in complex geometric domains: it does not require any geometric preprocessing or volumetric meshing, and its error decreases predictably with more samples. Yet, there remain many avenues for improvement.

To increase the efficiency of our estimator, we could search for subdomains other than star-shaped regions that allow larger steps while keeping throughput bounded. We could also develop: (1) better traversal strategies to quickly reject front- or back-facing geometry; (2) a high-performance wavefront (rather than megakernel) implementation for the GPU [Laine et al. 2013] that minimizes thread divergence amongst independent random walks of varying length; and (3) a single framework that combines the benefits of multiple variance reduction strategies such as boundary [Miller et al. 2023], mean value [Bakbouk and Peers 2023] or neural caching [Li et al. 2023], and bidirectional random walks [Qi et al. 2022].

Though we focus on solving Poisson equations, our estimator for Robin problems should apply more generally to elliptic PDEs with variable coefficients [Sawhney et al. 2022], and exterior problems with mixed boundary conditions [Nabizadeh et al. 2021]—here we can use the *Girsanov* and *Kelvin transforms* (*resp.*) to convert Neumann conditions into Robin conditions. Our algorithm can be used alongside a ray tracer to more accurately model physics that couples conduction, convection and radiative transfer [Bati et al. 2023]. We also believe that WoSt provides a valuable starting point for solving other PDEs with boundary integral formulations, such as the Helmholtz equation [Hunter and Pullan 2001, Chapter 3], linear elasticity [Hunter and Pullan 2001, Chapter 4] and even fluid flow [Busnello et al. 2005; Rioux-Lavoie et al. 2022]. Finally, similar to Monte Carlo rendering, another exciting direction for future work is to develop differentiable implementations of WoSt that optimize parametric descriptions of geometry and boundary conditions, for inverse tasks like electrical impedance tomography [Yılmazer et al. 2022] and thermally aware circuit board design [Cadence 2024].

ACKNOWLEDGMENTS

This work was supported by National Science Foundation (NSF) awards 2008123 and 2212290; National Institute of Food and Agriculture award 2023-67021-39073; a gift from Adobe Research; NSF Graduate Research Fellowship DGE2140739 and an NVIDIA graduate fellowship for Bailey Miller; a Packard Foundation Fellowship for Keenan Crane; and Alfred P. Sloan Research Fellowship FG202013153 for Ioannis Gkioulekas. Rohan Sawhney thanks Ken Museth for supporting this work. The Curiosity Mars rover model is courtesy of user *3d_molier International* on TurboSquid.

REFERENCES

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

James Arvo. 1995a. 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*. Springer, 115–126.

James Richard Arvo. 1995b. *Analytic methods for simulated light transport*. Ph. D. Dissertation. Yale University.

Ghada Bakbouk and Pieter Peers. 2023. Mean Value Caching for Walk on Spheres. In *Eurographics Symposium on Rendering*, Tobias Ritschel and Andrea Weidlich (Eds.). The Eurographics Association. https://doi.org/10.2312/sr.20231120

Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I. W. Levin, and Alec Jacobson. 2018. Fast Winding Numbers for Soups and Clouds. *ACM Trans. Graph.* 37, 4, Article 43 (2018), 12 pages.

Bruce A Barnes. 1990. Spectral properties of linear Volterra operators. *Journal of Operator Theory* (1990), 365–382.

Mégane Bati, Stéphane Blanco, Christophe Coustet, Vincent Eymet, Vincent Forest, Richard Fournier, Jacques Gautrais, Nicolas Mellado, Mathias Paulin, and Benjamin Piaud. 2023. Coupling Conduction, Convection and Radiative Transfer in a Single Path-Space: Application to Infrared Rendering. *ACM Trans. Graph.* 42, 4 (aug 2023).

Mireille Bossy, Nicolas Champagnat, Sylvain Maire, and Denis Talay. 2010. Probabilistic interpretation and random walk on spheres algorithms for the Poisson-Boltzmann equation in molecular dynamics. *ESAIM: Mathematical Modelling and Numerical Analysis* 44, 5 (2010), 997–1048.

Barbara Busnello, Franco Flandoli, and Marco Romito. 2005. 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 (2005), 295–336.

Cadence. 2024. Using a Thermal FEA Solver for Heat Management in Your PCB. Retrieved January 6, 2024 from https://resources.pcb.cadence.com/blog/2020-using-a-thermal-fea-solver-for-heat-management-in-your-pcb

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

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

Auguste De Lambilly, Gabriel Benedetti, Nour Rizk, Chen Hanqi, Siyuan Huang, Junnan Qiu, David Louapre, Raphael Granier De Cassagnac, and Damien Rohmer. 2023. Heat Simulation on Meshless Crafted-Made Shapes. In *Proceedings of the 16th ACM SIGGRAPH Conference on Motion, Interaction and Games* (<conf-loc>, <city>Rennes</city>, <country>France</country>, </conf-loc>) *(MIG '23)*. Association for Computing Machinery, New York, NY, USA, Article 9, 7 pages. https://doi.org/10.1145/3623264.3624457

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

SP Eveson. 2003. Norms of iterates of Volterra operators on L 2. *Journal of Operator Theory* (2003), 369–386.

Nicole Feng, Mark Gillespie, and Keenan Crane. 2023. Winding Numbers on Discrete Surfaces. *ACM Trans. Graph.* 42, 4, Article 36 (jul 2023), 17 pages. https://doi.org/10.1145/3592401

Natasha Flyer, Bengt Fornberg, Victor Bayona, and Gregory A Barnett. 2016. On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy. *J. Comput. Phys.* 321 (2016), 21–38.

Denis S Grebenkov. 2006. Partially reflected Brownian motion: a stochastic approach to transport phenomena. *arXiv preprint math/0610080* (2006).

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

Wolfgang Hackbusch. 2015. *Hierarchical matrices: algorithms and analysis*. Vol. 49. Springer Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-47324-5

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

Desmond J Higham. 2001. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM review* 43, 3 (2001), 525–546.

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4 (2020), 18 pages.

Peter Hunter and Andrew Pullan. 2001. Fem/bem notes. *Department of Engineering Science, The University of Auckland, New Zeland* (2001).

Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. 2013. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

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

Konrad Jörgens. 1982. *Linear integral operators*. Pitman.

Derek Juba, Walid Keyrouz, Michael Mascagni, and Mary Brady. 2016. Acceleration and Parallelization of ZENO/Walk-on-Spheres. *Procedia Computer Science* 80 (2016), 269–278. https://doi.org/10.1016/j.procs.2016.05.319 International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA.

Malvin H Kalos and Paula A Whitlock. 2009. *Monte carlo methods*. John Wiley & Sons.

Ehsan Kamel and Ali Kazemian. 2023. BIM-integrated thermal analysis and building energy modeling in 3D-printed residential buildings. *Energy and Buildings* 279 (2023), 112670.

Tosio Kato. 2013. *Perturbation theory for linear operators*. Vol. 132. Springer Science & Business Media.

Bastian Krayer and Stefan Müller. 2021. Hierarchical Point Distance Fields. In *International Symposium on Visual Computing*. Springer, 435–446.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. 2021. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 26548–26560.

Andreas E Kyprianou, Ana Osojnik, and Tony Shardlow. 2017. Unbiased 'walk-on-spheres' Monte Carlo methods for the fractional Laplacian. *IMA J. Numer. Anal.* 38, 3 (08 2017), 1550–1578. https://doi.org/10.1093/imanum/drx042 arXiv:https://academic.oup.com/imajna/article-pdf/38/3/1550/25170901/drx042.pdf

Eric P Lafortune and Yves D Willems. 1993. Bi-directional path tracing. In *Proc. Int. Conf. Comp. Graph. Vis. Tech.* Alvor, Portugal.

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

Shaofan Li and Wing Kam Liu. 2007. *Meshfree particle methods*. Springer Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-71471-2

Zilu Li, Guandao Yang, Xi Deng, Christopher De Sa, Bharath Hariharan, and Steve Marschner. 2023. Neural Caches for Monte Carlo Partial Differential Equation Solvers. In *SIGGRAPH Asia 2023 Conference Papers* (<conf-loc>, <city>Sydney</city>, <state>NSW</state>, <country>Australia</country>, </conf-loc>) *(SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 34, 10 pages. https://doi.org/10.1145/3610548.3618141

Frank Losasso, Ronald Fedkiw, and Stanley Osher. 2006. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids* 35, 10 (2006).

Michael Mascagni and Nikolai A Simonov. 2004. Monte Carlo Methods for Calculating Some Physical Properties of Large Molecules. *SIAM J. Sci. Comp.* 26, 1 (2004).

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2023. Boundary Value Caching for Walk on Spheres. *ACM Trans. Graph.* 42, 4, Article 82 (jul 2023), 11 pages. https://doi.org/10.1145/3592400

Jean-Paul Morillon. 1997. Numerical solutions of linear mixed boundary value problems using stochastic representations. *International journal for numerical methods in engineering* 40, 3 (1997), 387–405.

Linus Mossberg. 2021. *GPU-Accelerated Monte Carlo Geometry Processing for Gradient-Domain Methods*. Ph. D. Dissertation. Linköping University, Linköping, Sweden.

Mervin E Muller. 1956. Some Continuous Monte Carlo Methods for the Dirichlet Problem. *Annals of Mathematical Statistics* 27, 3 (Sept. 1956), 569–589.

Mohammad Sina Nabizadeh, Ravi Ramamoorthi, and Albert Chern. 2021. Kelvin Transformations for Simulations on Infinite Domains. *ACM Trans. Graph.* 40, 4, Article 97 (jul 2021), 15 pages. https://doi.org/10.1145/3450626.3459809

NASA. 1999. Guidelines for Thermal Analysis of Spacecraft Hardware. Retrieved January 6, 2024 from https://llis.nasa.gov/lesson/695

Jean-Claude Nédélec. 2001. *Acoustic and electromagnetic equations: integral representations for harmonic problems*. Vol. 144. Springer.

B. Øksendal. 2003. Stochastic Differential Equations: An Introduction with Applications. Springer Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14394-6

Paul William Partridge, Carlos Alberto Brebbia, et al. 2012. *Dual reciprocity boundary element method*.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

Yang Qi, Dario Seyb, Benedikt Bitterli, and Wojciech Jarosz. 2022. A bidirectional formulation for Walk on Spheres. *Computer Graphics Forum* 41, 4 (2022), 51–62. https://doi.org/10.1111/cgf.14586 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14586

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics* 378 (2019), 686–707.

Damien Rioux-Lavoie, Ryusuke Sugimoto, Tümay Özdemir, Naoharu H. Shimada, Christopher Batty, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2022. A Monte Carlo Method for Fluid Simulation. *ACM Trans. Graph.* 41, 6, Article 240 (nov 2022), 16 pages. https://doi.org/10.1145/3550454.3555450

Karl K Sabelfeld and Nikolai A Simonov. 2013. *Random walks on boundary for solving PDEs*. De Gruyter.

Karl K Sabelfeld and Nikolai A Simonov. 2016. *Stochastic methods for boundary value problems: numerics for high-dimensional PDEs and applications*. Walter de Gruyter GmbH & Co KG.

Rohan Sawhney. 2021. *FCPW: Fastest Closest Points in the West*. https://github.com/rohan-sawhney/fcpw

Rohan Sawhney and Keenan Crane. 2020. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Trans. Graph.* 39, 4, Article 123 (aug 2020), 18 pages. https://doi.org/10.1145/3386569.3392374

Rohan Sawhney and Bailey Miller. 2023. *Zombie: A Grid-Free Monte Carlo Solver for PDEs*. https://github.com/rohan-sawhney/zombie

Rohan Sawhney, Bailey Miller, Ioannis Gkioulekas, and Keenan Crane. 2023. Walk on Stars: A Grid-Free Monte Carlo Method for PDEs with Neumann Boundary Conditions. *ACM Trans. Graph.* 42, 4, Article 80 (jul 2023), 20 pages. https://doi.org/10.1145/3592398

Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. 2022. Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients. *ACM Trans. Graph.* 41, 4, Article 53 (jul 2022), 17 pages. https://doi.org/10.1145/3528223.3530134

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (feb 2015), 36 pages. https://doi.org/10.1145/2629697

Nikolai A Simonov. 2017. Walk-on-spheres algorithm for solving third boundary value problem. *Applied Mathematics Letters* 64 (2017), 156–161.

Cyril Soler, Ronak Molazem, and Kartic Subr. 2022. A Theoretical Analysis of Compactness of the Light Transport Operator. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 17, 9 pages. https://doi.org/10.1145/3528233.3530725

Ryusuke Sugimoto, Terry Chen, Yiti Jiang, Christopher Batty, and Toshiya Hachisuka. 2023. A Practical Walk-on-Boundary Method for Boundary Value Problems. *ACM Trans. Graph.* 42, 4 (aug 2023), 16 pages. https://doi.org/10.1145/3592109

Eric Veach. 1998. *Robust Monte Carlo methods for light transport simulation*. Stanford University.

Eric Veach and Leonidas Guibas. 1995. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*. Springer, 145–167.

Alan V Von Arx and Adon Delgado Jr. 1991. Convective heat transfer on Mars. In *AIP Conference Proceedings*, Vol. 217. American Institute of Physics, 734–739.

Ingo Wald. 2007. On fast construction of SAH-based bounding volume hierarchies. In *2007 IEEE Symposium on Interactive Ray Tracing*. IEEE, 33–40.

Ekrem Fatih Yılmazer, Delio Vicini, and Wenzel Jakob. 2022. Solving Inverse PDE Problems using Grid-Free Monte Carlo Estimators. *arXiv preprint arXiv:2208.02114* (2022).

Yijing Zhou and Wei Cai. 2016. Numerical Solution of the Robin Problem of Laplace Equations with a Feynman–Kac Formula and Reflecting Brownian Motions. *Journal of Scientific Computing* 69 (2016). https://doi.org/10.1007/s10915-016-0184-y

A OPERATOR-THEORETIC ANALYSIS

We provide a formal justification for the convergence and bounded variance of WoSt with Dirichlet, Neumann and Robin conditions, via an operator-theoretic analysis of boundary integral equations. We draw inspiration from the seminal work of Arvo [1995a], who pioneered the use of operator theory to analyze the convergence of integral equations for light transport—Arvo [1995b, Chapter 6] and Veach [1998, Chapter 4 & 7] provide a detailed treatment of this topic, and Soler et al. [2022] describe recent developments.

Background. The material in this section uses results from Jörgens [1982, Chapter 2] and Kato [2013, Chapter 3]. To simplify the discussion, we work with a Banach space $\mathcal{L}^2(\Omega)$ of square-integrable functions on a domain Ω, equipped with the usual functional norm $\|\cdot\|_{\mathcal{L}^2(\Omega)}$. That is, for any function $u \in \mathcal{L}^2(\Omega)$,

$$\|u\|_{\mathcal{L}^2(\Omega)} := \left(\int_\Omega |u(x)|^2 \, \mathrm{d}x\right)^{\frac{1}{2}} \text{ and } \|u\|_{\mathcal{L}^2(\Omega)} < \infty. \tag{16}$$

Even though the solution of linear elliptic PDEs such as the Poisson equation generally exists 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* $\kappa : \Omega \times \Omega \to \mathbb{R}$ and a *boundary mapping* $\mathrm{A} : \Omega \to \mathcal{P}(\Omega)$ (where $\mathcal{P}$ is the powerset), we define an integral operator

$$C_\kappa^{\mathrm{A}}[u](x) := \int_{\mathrm{A}(x)} \kappa(x,y) u(y) \, \mathrm{d}y. \tag{17}$$

If the kernel and boundary mapping satisfy, for some constant C,

$$\sup_{x\in\Omega} \int_{\mathrm{A}(x)} |\kappa(x,y)| \, \mathrm{d}y < C < \infty, \tag{18}$$

then the integral operator C_κ^{A} satisfies the following properties:

(1) Its codomain is $\mathcal{L}^2(\Omega)$, *i.e.*, $C_\kappa^{\mathrm{A}} : \mathcal{L}^2(\Omega) \to \mathcal{L}^2(\Omega)$.
(2) It is bounded, and in particular, for all $u \in \mathcal{L}^2(\Omega)$,

$$\left\|C_\kappa^{\mathrm{A}}[u]\right\|_{\mathcal{L}^2(\Omega)} \le C\|u\|_{\mathcal{L}^2(\Omega)}. \tag{19}$$

These two properties together imply that the operator has a well defined operator norm

$$\left\|C_\kappa^{\mathrm{A}}\right\|_{\mathrm{Op}} := \sup\left\{\frac{\left\|C_\kappa^{\mathrm{A}}[u]\right\|_{\mathcal{L}^2(\Omega)}}{\|u\|_{\mathcal{L}^2(\Omega)}}, u \in \mathcal{L}^2(\Omega), u \neq 0\right\}, \tag{20}$$

and spectral radius

$$r(C_\kappa^{\mathrm{A}}) := \lim_{n\to\infty} \left\|C_\kappa^{\mathrm{A}}\right\|_{\mathrm{Op}}^{\frac{1}{n}}. \tag{21}$$

It also follows that

$$\left\|C_\kappa^{\mathrm{A}}\right\|_{\mathrm{Op}} \le C \text{ and } r(C_\kappa^{\mathrm{A}}) < C. \tag{22}$$

Moreover, for an operator C_κ^{A}, its unit resolvent is the operator

$$\mathcal{R}_\kappa^{\mathrm{A}} := \left(\mathcal{I} - C_\kappa^{\mathrm{A}}\right)^{-1}, \tag{23}$$

where $\mathcal{I} : \mathcal{L}^2(\Omega) \to \mathcal{L}^2(\Omega)$ is the identity operator. The unit resolvent $\mathcal{R}_\kappa^{\mathrm{A}}$ exists and is a bounded operator $\mathcal{L}^2(\Omega) \to \mathcal{L}^2(\Omega)$ if and only if $r(C_\kappa^{\mathrm{A}}) < 1$ (the inequality is strict).

WoSt with Dirichlet-Neumman conditions. To understand the convergence of WoSt with mixed Dirichlet-Neumann conditions, we will express the boundary integral in Equation 3 in operator-theoretic form. We achieve this by defining the convolutional kernel $\kappa := P^{\mathrm{B}}$, the boundary mapping $\mathrm{A} : x \mapsto \partial\mathrm{St}(x,R)$, and the corresponding integral operator C_P^{St}—we leave the radius R unspecified for now, which means that the region $\mathrm{St}(x,R)$ is not necessarily star-shaped. We can then rewrite Equation 3 equivalently in operator form as

$$u = C_P^{\mathrm{St}}[u] + s, \tag{24}$$

where we use $s \in \mathcal{L}^2(\Omega)$ to denote the non-recursive terms on the right-hand side of Equation 3. Equation 24 is a Fredholm-Volterra equation of the second kind, whose solution exists only if the unit resolvent $\mathcal{R}_P^{\mathrm{St}} := (\mathcal{I} - C_P^{\mathrm{St}})^{-1}$ of C_P^{St} exists and is bounded [Barnes 1990; Eveson 2003]. If $\mathcal{R}_P^{\mathrm{St}}$ is indeed bounded, then by rearranging terms we can write the solution as

$$u = \mathcal{R}_P^{\mathrm{St}}[s]. \tag{25}$$

To estimate Equation 24 using recursive Monte Carlo, we also require boundedness of $\mathcal{R}_P^{\mathrm{St}}$ for any estimator to be convergent and have finite variance [Heinrich and Mathé 1993]. Therefore, to ensure the WoSt estimator has these properties, we need to select a radius R for region $\mathrm{St}(x,R)$ that guarantees this condition on $\mathcal{R}_P^{\mathrm{St}}$.

To this end, we leverage the fact that the Poisson kernel P^{B} is the *signed solid angle* kernel that integrates to 1 over any closed region [Jacobson et al. 2013; Barill et al. 2018; Feng et al. 2023]. Thus

$$\int_{\partial\mathrm{St}(x,R)} \left|P^{\mathrm{B}}(x,y)\right| dy \ge \int_{\partial\mathrm{St}(x,R)} P^{\mathrm{B}}(x,y) dy = 1. \tag{26}$$

The inequality follows from basic properties of the absolute value and integration, and becomes an equality in regions $\mathrm{St}(x,R)$ where P^{B} 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 distances 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

$$\int_{\partial\mathrm{St}(x,R)} \left|P^{\mathrm{B}}(x,y)\right| dy = 1. \tag{27}$$

Together with Equation 22, Equation 27 guarantees that the spectral radius $r(C_P^{\mathrm{St}}) \le 1$. However, for the unit resolvent to exist and be bounded, we require this inequality to be strict. We achieve this through the ε-shell approximation, which effectively replaces P^{B} 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,

$$\int_{\partial\mathrm{St}(x,R)} \left|P_\varepsilon^{\mathrm{B}}(x,y)\right| dy < \int_{\partial\mathrm{St}(x,R)} \left|P^{\mathrm{B}}(x,y)\right| dy = 1, \tag{28}$$

ensuring the unit resolvent exists. As WoSt reduces to WoS for pure Dirichlet problems, this analysis also applies to WoS, whose spherical domains always satisfy Equation 27—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 [Sawhney et al. 2023, Section 3.4.3], as the lack of an ε-shell (only defined on Dirichlet boundaries) means that $P^{\mathrm{B}}_{\varepsilon}(x,y) = P^{\mathrm{B}}(x,y)$ always, and thus the spectral radius $r(C^{\mathrm{St}}_{P})$ is not guaranteed to be smaller than 1.

WoSt with Robin conditions. Lastly, we consider WoSt with Robin conditions (Section 4). As above, we start by writing the boundary integral in Equation 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 $C^{\mathrm{St}}_{\rho_\mu P}$, we rewrite Equation 6 equivalently in operator form as

$$u = C^{\mathrm{St}}_{\rho_\mu P}[u] + s, \tag{29}$$

As in the previous section, we need to select a radius R that ensures the unit resolvent $\mathcal{R}^{\mathrm{St}}_{\rho_\mu P} := (\mathcal{I} - C^{\mathrm{St}}_{\rho_\mu P})^{-1}$ of $C^{\mathrm{St}}_{\rho_\mu P}$ exists and is bounded. This will guarantee that the solution

$$u = \mathcal{R}^{\mathrm{St}}_{\rho_\mu P}[s] \tag{30}$$

can be estimated using recursive Monte Carlo. Using the radius for a Dirichlet-Neumann problem, we have

$$\int_{\partial \mathrm{St}(x,R)} \left|\rho_\mu(x,y) P^{\mathrm{B}}(x,y)\right| dy \le \int_{\partial \mathrm{St}(x,R)} \left|\rho_\mu(x,y)\right| dy \int_{\partial \mathrm{St}(x,R)} \left|P^{\mathrm{B}}(x,y)\right| dy = \int_{\partial \mathrm{St}(x,R)} \left|\rho_\mu(x,y)\right| dy, \tag{31}$$

where we used Hölder's inequality and Equation 27. To ensure that

$$\int_{\partial \mathrm{St}(x,R)} \left|\rho_\mu(x,y)\right| dy \le 1, \tag{32}$$

we follow Section 4.2 to further restrict R such that $|\rho_\mu(x,y)| \le 1$ 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 (Section 4.3), which is not possible when the boundary conditions are Dirichlet and Neumann.

B REFLECTANCE AND RADIUS BOUND IN 2D

To obtain an explicit expression for the reflectance in 2D, we substitute the 2D Green's function and Poisson kernel of a ball $\mathrm{B}(x,R)$ [Sawhney et al. 2023, Equations 25 & 26] into Equation 7 to obtain

$$\rho_\mu(x,z) = 1 - \frac{\mu(z)\, r}{\cos\theta} \left(\log(R) - \log(r)\right), \tag{33}$$

where $r = \|z - x\|$ and $\cos\theta = {(n_z \cdot (z-x))}/{r}$. To restrict $\rho_\mu \in [0,1]$, we require

$$\frac{\mu(z)\, r}{\cos\theta} \left(\log(R) - \log(r)\right) \le 1. \tag{34}$$

Then, rearranging terms gives us an upper bound on the radius R,

$$R \le r \exp\left(\frac{\cos\theta}{\mu(z)\, r}\right), \tag{35}$$

which must hold at *all* points $z \in \partial \mathrm{St}_{\mathrm{R}}$.

Similar to Section 5.1, 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 Equation 35, we have:

$$R \le \frac{h}{\cos\theta} \exp\left(\frac{\cos^2\theta}{\mu(z)\, h}\right). \tag{36}$$

As before, we minimize this equation with respect to $\cos\theta$. This gives us an analytic expression $\sqrt{\mu^{\max} h/2}$ for the cosine, which we clamp between the minimum and maximum cosine values at the closest and farthest points on l (*resp.*). We then plug the resulting cosine value back in Equation 36 to compute the radius bound for l.

C USING WOST TO ESTIMATE G^{Ω}

In Section 6.2, G^{Ω} satisfies a Poisson equation with Robin boundary conditions (Equation 15). To use WoSt to estimate G^{Ω}, we therefore follow Section 4.1 to derive an integral expression on a star-shaped region St. For any two points $x \in \Omega$ and $z \in \partial\Omega_{\mathrm{R}}$, we have

$$G^{\Omega}(x,z) = \int_{\partial \mathrm{St}(x,R)} \rho_\mu(x,z') P^{\mathrm{B}}(x,z') G^{\Omega}(z',z)\, \mathrm{d}z' + G^{\mathrm{B}}(x,z) \tag{37}$$

$$= \int_{\partial \mathrm{St}(z,R)} \rho_\mu(z,z') P^{\mathrm{B}}(z,z') G^{\Omega}(z',x)\, \mathrm{d}z' + G^{\mathrm{B}}(z,x), \tag{38}$$

where the second equality follows from G^{Ω} being symmetric. We use this expression to define random walks that start from randomly sampled points on $\partial\Omega_{\mathrm{R}}$ with known Robin data h; for the source term f, walks will instead start from Ω. However, unlike the "forward" estimator from Equation 8 which gathers information about boundary conditions and source values during a walk, a "reverse" WoSt estimator for G^{Ω} will instead allow for a single walk to splat h and f values at more than one point inside Ω (similar to the reverse WoS walks of Qi et al. [2022] for pure Dirichlet problems). In more detail, for $k \ge 0$, we estimate G^{Ω} using the following recursive single-sample estimator for Equation 38:

$$\widehat{G}^{\Omega}(x, z_k) = \rho_\mu(z_k, z_{k+1}) \widehat{G}^{\Omega}(z_{k+1}, x) \ + \ G^{\mathrm{B}}(z_k, x). \tag{39}$$

Here $G^{\mathrm{B}}(z_k, x) > 0$ when $x \in \mathrm{St}(z_k, R_k)$, and is zero otherwise. As in Section 3.2.3, we use direction sampling to determine the next walk location $z_{k+1} \in \partial\mathrm{St}(z_k, R_k)$. As in Section 4.3, we use Russian roulette to terminate walks with probability $\rho_\mu(z_k, z_{k+1})$.

We then estimate the BVP solution in Equation 14 as

$$\widehat{u}(x) = -\frac{\widehat{G}^{\Omega}(x, z_0)\, h(z_0)}{\alpha(x)\, p^{\partial\Omega_{\mathrm{R}}}(z_0)} + \frac{\widehat{G}^{\Omega}(x, y_0)\, f(y_0)}{\alpha(x)\, p^{\Omega}(y_0)}, \tag{40}$$

where we are free to sample points $z_0 \in \partial\Omega_{\mathrm{R}}$ and $y_0 \in \Omega_{\mathrm{R}}$ from densities $p^{\partial\Omega_{\mathrm{R}}}$ and p^{Ω} (*resp.*) of our choosing. Finally, we use Equation 40 and the random walks from Equation 39 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 $\mu = 0$.

D PSEUDOCODE

Algorithms 2, 3 and 4 provide pseudocode for the geometric queries we describe in Section 5.

ALGORITHM 2: StarRadiusReflectingBoundary(T = Snch(triangles$_{\partial\Omega_R}$), x, R, $d_T^{\min} = 0$)
Note: Code annotated with comments in green indicates our modifications to the corresponding procedure in Sawhney et al. [2023].

```
Input: Spatialized normal cone hierarchy $T$, query point $x \in \mathbb{R}^3$, current radius estimate $R$, and minimum distance $d_T^{\min}$ to $T$'s aabb from $x$ (0 if $x \in$ aabb).
Output: Radius of star-shaped region $\mathrm{St}(x)$ containing a portion of the reflecting boundary $\partial\Omega_R$.
1: if $d_T^{\min} > R$ then return $R$    ▷Ignore nodes outside current radius estimate
2: if $T$.isLeaf then
3:     for $t$ in $T$.triangles do
4:         $R \leftarrow$ StarRadiusTriangle($t$, $x$, $R$)    ▷Compute radius bound for triangle $t$ (Alg. 3)
5: else
6:     visitLeft, $R_{\text{left}}^{\min}$, $R_{\text{left}}^{\max} \leftarrow$ VisitNode($T$.left, $x$, $R$)    ▷Determine whether to visit left node (Alg. 4)
7:     if visitLeft then $R \leftarrow \min(R, R_{\text{left}}^{\max})$
8:     visitRight, $R_{\text{right}}^{\min}$, $R_{\text{right}}^{\max} \leftarrow$ VisitNode($T$.right, $x$, $R$)    ▷Determine whether to visit right node (Alg. 4)
9:     if visitRight then $R \leftarrow \min(R, R_{\text{right}}^{\max})$
10:    if visitLeft and visitRight then
11:        if $R_{\text{left}}^{\min} < R_{\text{right}}^{\min}$ then    ▷Visit closer node first
12:            $R \leftarrow$ StarRadiusReflectingBoundary($T$.left, $x$, $R$, $R_{\text{left}}^{\min}$)
13:            $R \leftarrow$ StarRadiusReflectingBoundary($T$.right, $x$, $R$, $R_{\text{right}}^{\min}$)
14:        else
15:            $R \leftarrow$ StarRadiusReflectingBoundary($T$.right, $x$, $R$, $R_{\text{right}}^{\min}$)
16:            $R \leftarrow$ StarRadiusReflectingBoundary($T$.left, $x$, $R$, $R_{\text{left}}^{\min}$)
17:    else if visitLeft then $R \leftarrow$ StarRadiusReflectingBoundary($T$.left, $x$, $R$, $R_{\text{left}}^{\min}$)    ▷Visit only left node
18:    else if visitRight then $R \leftarrow$ StarRadiusReflectingBoundary($T$.right, $x$, $R$, $R_{\text{right}}^{\min}$)    ▷Visit only right node
19: return $R$
```

ALGORITHM 3: StarRadiusTriangle(t, x, $R^{\max} = \infty$)
Note: Code annotated with comments in green indicates our modifications to the corresponding procedure in Sawhney et al. [2023].

```
Input: Triangle $t$ with min and max robin coefficients $\mu^{\min}$ and $\mu^{\max}$, query point $x \in \mathbb{R}^3$, and radius bound $R^{\max}$.
Output: Radius of star-shaped region for triangle $t$.
1: $R_t \leftarrow R^{\max}$    ▷Initialize radius value
2: $d^{\text{closest}}$, $x^{\text{closest}} \leftarrow$ ClosestPointTriangle($t$, $x$)
3: if $d^{\text{closest}} > R_t$ then return $R_t$    ▷$t$ is outside radius bound, return the radius bound
4: if $t.\mu^{\max} \equiv \infty$ then return $d^{\text{closest}}$    ▷$t$ has Dirichlet conditions, return distance to closest point on $t$
5: $n_t \leftarrow$ TriangleNormal($t$)
6: for $e$ in $t$.adjacentEdges do    ▷Visit edges adjacent to $t$ and compute distance to closest silhouette edge
7:     $p_e \leftarrow$ ClosestPointEdge($e$, $x$)
8:     $v \leftarrow p_e - x$
9:     $d_e \leftarrow |v|$
10:    if $d_e < R_t$ then
11:        hasAdjacentTriangle, $n_{\text{adj}} \leftarrow$ AdjacentTriangleNormal($t$, $e$)
12:        isSilhouetteEdge $\leftarrow$ not hasAdjacentTriangle or $(v \cdot n_t) \cdot (v \cdot n_{\text{adj}}) \le 0$
13:        if isSilhouetteEdge then $R_t \leftarrow \min(R_t, d_e)$
14: if $t.\mu^{\min} \equiv 0$ then return $R_t$    ▷$t$ has Neumann conditions, return distance to closest silhouette edge
15: else    ▷$t$ has Robin conditions, compute radius bound for $t$ (Eq. 12)
16:    $d^{\text{farthest}}$, $x^{\text{farthest}} \leftarrow$ FarthestPointTriangle($t$, $x$)
17:    $\cos^{\max}\theta \leftarrow |n_t \cdot (x - x^{\text{farthest}})| \,/\, d^{\text{farthest}}$
18:    $\cos^{\min}\theta \leftarrow |n_t \cdot (x - x^{\text{closest}})| \,/\, d^{\text{closest}}$
19:    $h \leftarrow$ DistancePlane($x^{\text{closest}}$, $n_t$)
20:    $\mu h \leftarrow t.\mu^{\max} \cdot h$
21:    if $\sqrt{\mu h} < \cos^{\min}\theta$ then return $R_t$
22:    $\cos\theta \leftarrow$ Clamp($\sqrt{\mu h \,/\, 3}$, $\cos^{\min}\theta$, $\cos^{\max}\theta$)
23:    return $\min\left(R_t, \frac{\mu h^2}{\mu h \cos\theta - \cos^3\theta}\right)$
```

ALGORITHM 4: VisitNode$(T,\ x,\ R = \infty)$
Note: Code annotated with comments in green indicates our modifications to the corresponding procedure in Sawhney et al. [2023].

```
Input: Spatialized normal cone hierarchy node $T$ with min and max robin coefficients $\mu^{\min}$ and $\mu^{\max}$, query point $x \in \mathbb{R}^3$, and current radius estimate $R$.
Output: Whether to traverse the node, as well as min and max bounds on the star radius computed using $T$'s aabb and cone.
1: visit, $d^{\min}$, $d^{\max} \leftarrow$ IntersectAABBSphere($T$.aabb, $x$, $R$)    ▷ Intersect aabb with sphere $\partial\mathrm{B}(x, R)$, and compute min and max distance to aabb from $x$
2: if not visit then return false, $0, \infty$    ▷ Do not visit node as aabb does not intersect $\mathrm{B}(x, R)$
3: if $T.\mu^{\min} \equiv \infty$ then return true, $d^{\min}, d^{\max}$    ▷ $T$ only contains a Dirichlet boundary, visit node
4: hasSilhouette, $|\cos^{\min}\theta|$, $|\cos^{\max}\theta| \leftarrow$ HasSilhouette($T$.aabb, $T$.cone, $x$)    ▷ Sawhney et al. [2023, Alg. 4]
5: if hasSilhouette then return true, $d^{\min}, \infty$    ▷ Visit node since the normal & view cones formed by aabb & $x$ likely contain a pair of orthogonal directions
6: else if $T.\mu^{\max} \equiv 0$ then return false, $0, \infty$    ▷ Do not visit node as $T$ contains an entirely front- or back-facing Neumann boundary with no silhouette edge
7: $R^{\min} \leftarrow d^{\max} \leq \frac{|\cos^{\min}\theta|}{T.\mu^{\max}}\ ?\ \infty\ :\ d^{\min} \Big/ \left(1 - \frac{|\cos^{\min}\theta|}{T.\mu^{\max}\ d^{\max}}\right)$    ▷ Compute minimum radius bound for Robin boundary inside T (Eq. 11)
8: $R^{\max} \leftarrow d^{\min} \leq \frac{|\cos^{\max}\theta|}{T.\mu^{\min}}\ ?\ \infty\ :\ d^{\max} \Big/ \left(1 - \frac{|\cos^{\max}\theta|}{T.\mu^{\min}\ d^{\min}}\right)$    ▷ Compute maximum radius bound for Robin boundary inside T (Eq. 11)
9: return true, $R^{\min}, R^{\max}$    ▷ Visit node
```