Boundary Value Caching for Walk on Spheres

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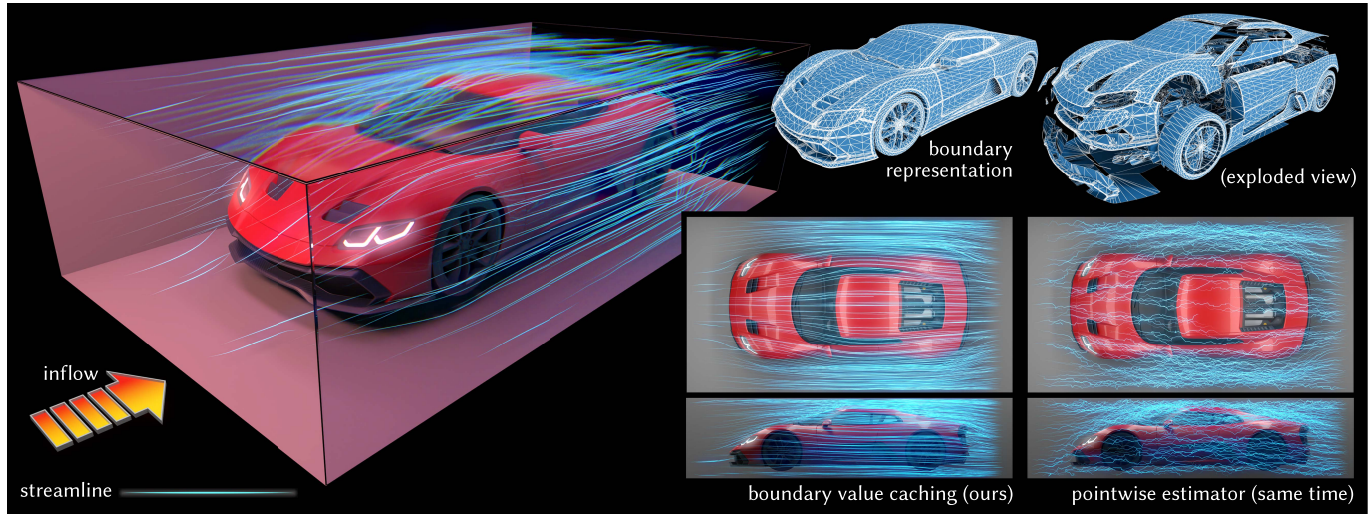IOANNIS GKIOULEKAS†, Carnegie Mellon University, USA

Fig. 1. Our caching scheme dramatically reduces the total number of random walks needed to solve partial differential equations relative to classic pointwise Monte Carlo estimators. Here we show streamlines of a flow in a simulated wind tunnel, computed directly from a low-quality surface mesh originally intended for visualization rather than simulation.

Grid-free Monte Carlo methods such as *walk on spheres* can be used to solve elliptic partial differential equations without mesh generation or global solves. However, such methods independently estimate the solution at every point, and hence do not take advantage of the high spatial regularity of solutions to elliptic problems. We propose a fast caching strategy which first estimates solution values and derivatives at randomly sampled points along the boundary of the domain (or a local region of interest). These cached values then provide cheap, output-sensitive evaluation of the solution (or its gradient) at interior points, via a boundary integral formulation. Unlike classic boundary integral methods, our caching scheme introduces zero statistical bias and does not require a dense global solve. Moreover we can handle imperfect geometry (e.g., with self-intersections) and detailed boundary/source terms without repairing or resampling the boundary representation. Overall, our scheme is similar in spirit to *virtual point light* methods from photorealistic rendering: it suppresses the typical salt-and-pepper noise characteristic of independent Monte Carlo estimates, while still retaining the many advantages of Monte Carlo solvers: progressive evaluation, trivial parallelization, geometric robustness, *etc.* We validate our approach using test problems from visual and geometric computing.

---

*and † indicate equal contribution.

Authors' addresses: Bailey Miller, bmmiller@andrew.cmu.edu, Carnegie Mellon University, USA; Rohan Sawhney, rohansawhney@cs.cmu.edu, Carnegie Mellon University, USA and NVIDIA, USA; Keenan Crane, kmcrane@cs.cmu.edu, Carnegie Mellon University, USA; Ioannis Gkioulekas, igkioule@cs.cmu.edu, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA.

1 INTRODUCTION

The *walk on spheres (WoS)* method solves problems like the Laplace or Poisson equation by aggregating information from repeated random walks [Muller 1956; Sawhney and Crane 2020]. Like Monte Carlo ray tracing—and unlike conventional partial differential equation (PDE) solvers—it does not require a mesh of the problem domain, nor even a high-quality mesh of its boundary. This fact makes WoS valuable for problems in visual and geometric computing, as one can directly use imperfect assets from design or visualization to perform simulation and analysis (Figure 1). However, classic WoS methods estimate the PDE solution *pointwise* and do not share information between sample points, resulting in highly redundant computation.

We propose a simple *boundary value caching (BVC)* scheme, well-suited for problems like visualization, where the solution must be evaluated densely in space. This scheme is enabled by the recent *walk on stars (WoSt)* method [Sawhney et al. 2023], which extends WoS to problems with mixed Neumann and Dirichlet boundary conditions. In particular, we consider PDEs of the form

$$\begin{aligned} \Delta u - \sigma u &= f && \text{on } \Omega \\ u &= g && \text{on } \partial\Omega_D \\ \frac{\partial u}{\partial n} &= h && \text{on } \partial\Omega_N \end{aligned} \quad (1)$$

where the boundary of the domain $\Omega \subset \mathbb{R}^n$ is split into a Dirichlet part $\partial\Omega_D$ and Neumann part $\partial\Omega_N$ with prescribed values g and derivatives h respectively. Here Δ is the negative-semidefinite Laplacian, $\sigma \in \mathbb{R}_{\geq 0}$ is a constant, and f is a given source term. At interior points x, the solution to Equation (1) is given by the *boundary integral equation (BIE)* [Costabel 1987; Hunter and Pullan 2001]

$$u(x) = \underbrace{\int_{\partial\Omega} \frac{\partial G}{\partial n}(x, z)u(z) - G(x, z)\frac{\partial u}{\partial n}(z)\, \mathrm{d}z}_{=:u_{\partial\Omega}(x)} + \underbrace{\int_{\Omega} G(x, y)\, f(y)\, \mathrm{d}y}_{=:u_{\Omega}(x)}, \quad (2)$$

where G is the free-space Green's function for Equation (1), and n is the unit outward normal at the boundary.

To make use of the BIE, one must somehow determine the unknown boundary data: Dirichlet values u on the Neumann boundary $\partial\Omega_N$, and Neumann values $\partial u/\partial n$ on the Dirichlet boundary $\partial\Omega_D$. Schemes such as the *boundary element method (BEM)* use a finite-dimensional space of functions on the boundary (*e.g.*, basis functions associated with mesh nodes), and solve a dense, globally-coupled linear system for the best approximation to the true solution.

We take a completely different approach, and instead use random walks to compute the unknown boundary values. In particular, we use WoS(t) to obtain u along $\partial\Omega_N$ and $\partial u/\partial n$ along $\partial\Omega_D$ (Section 3.1). This approach avoids global solves, boundary remeshing, and approximation of the function space; unlike BEM, it also handles the source term f. Moreover, as random walks can be expensive (especially in problems with predominantly Neumann boundaries), we cache these boundary values at a collection of random sample points along $\partial\Omega$. We can then use a Monte Carlo estimate of Equation (2) to cheaply evaluate the solution at any interior point x, without taking any further random walks (Algorithm 1). This scheme is easy to parallelize, and can be computed progressively (*e.g.*, for interactive preview). We can also focus computation on a region of interest by caching points only on the boundary of a small subdomain $R \subset \Omega$ (Section 3.2)—unlike BEM which must always perform a global solve involving the entire boundary $\partial\Omega$.

In practice, we obtain far smoother results across the domain with our method compared to directly using pointwise estimators like WoS or WoSt (Figures 1, 5 and 7). This behavior can be attributed to correlations in the solution estimates at interior evaluation points that use the same boundary and source samples. On the flip side, error is now more global akin to traditional PDE solvers such as FEM and BEM (Figure 12). Unlike pointwise estimators, we also observe boundary artifacts (Figure 11) as samples are no longer generated in proportion to the singular functions G and $\partial G/\partial n$. We show how to mitigate such artifacts in Section 3.3.
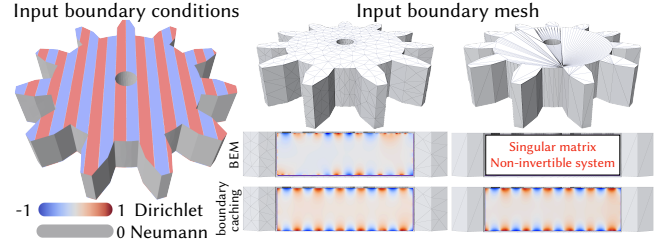
Fig. 2. Even for problems with relatively simple boundary conditions and no source term (*left*), finite element technology like BEM suffers from large global errors in the PDE solution without significant mesh refinement due to local aliasing of boundary data, and can fail completely on domains with irregular elements (*middle row*). In contrast, our method solves PDEs without any aliasing artifacts irrespective of tesselation quality as it decouples problem inputs from the boundary representation (*bottom row*).

2 RELATED WORK

Though we exclusively employ a grid-free Monte Carlo approach to solve PDEs such as the Poisson equation, numerous deterministic and stochastic solvers exist to accomplish this task with varying tradeoffs. We refer the reader to Sawhney and Crane [2020, Section 7] and Sawhney et al. [2022, Section 7] for a comprehensive overview of the advantages of a Monte Carlo approach like ours over traditional grid-based methods such as finite differences (FD), finite elements (FEM) and "meshless" finite elements (MFEM). Here we instead focus on techniques that directly evaluate BIEs, as well as sample reuse strategies in rendering our method is inspired by.

Grid-free Monte Carlo methods. Originally developed by Muller [1956], WoS was recently introduced to graphics by Sawhney and Crane [2020] who made the link to techniques from Monte Carlo rendering. It has since been generalized to handle Neumann boundary conditions [Sawhney et al. 2023], solve variable coefficient PDEs [Sawhney et al. 2022], simulate fluids [Rioux-Lavoie et al. 2022] and solve inverse problems via a differentiable formulation [Yılmazer et al. 2022]. However, apart from proof-of-concept demonstrations [Sawhney and Crane 2020, Figures 12 & 13], a practical scheme for denoising results has yet to be developed. Qi et al. [2022]'s recent bidirectional WoS formulation provides substantial variance reduction in problems with concentrated source terms, but is currently limited to only Dirichlet conditions and is not output sensitive—in contrast, our method can handle mixed boundary conditions and provides explicit control over the location and number of cached samples assigned to each evaluation point for estimation. As in rendering [Segovia et al. 2006], it is likely that Qi et al. [2022]'s bidirectional approach can be used within our caching strategy to also estimate the solution at boundary samples.

Deterministic Boundary Element Solvers. Much like our grid-free PDE solver, the boundary element method (BEM) is designed to solve BIEs and does not discretize the interior of the domain. However, there are significant differences in capabilities: to evaluate Equation (2), BEM must first discretize the boundary geometry, leading to spatial aliasing in the boundary data (Figure 2). It must then invert a *dense* linear system of equations for the unknown solution value u
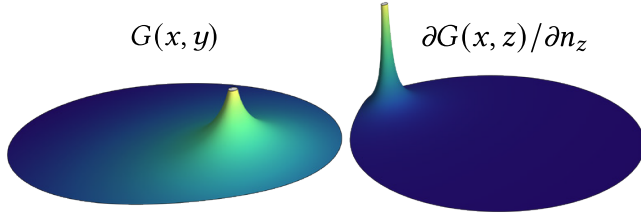
Fig. 3. The free-space Green's function and its normal derivative in Equation (2) are singular at the point they are centered on, but decay smoothly and fall-off quickly away from the singularity.

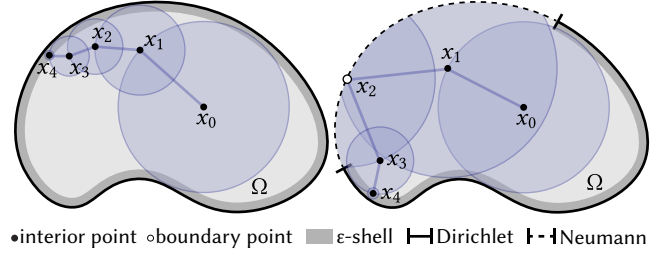•interior point ∘boundary point ▇ε-shell ⊢⊣Dirichlet ⊢-⊣Neumann

Fig. 4. *Left:* Walk on spheres simulates a *Brownian* random walk inside a Dirichlet boundary $\partial\Omega_D$ to solve elliptic PDEs, repeatedly jumping to a random point on the largest sphere centered at the current walk location. The walk is terminated when it enters an epsilon shell $\partial\Omega_D^\epsilon$ around the boundary. *Right:* Walk on stars generalizes WoS to domains with a Neumann boundary $\partial\Omega_N$, using a sphere that can contain a subset of $\partial\Omega_N$ inside it. The next walk location is determined by intersecting a ray with a uniformly sampled direction against the sphere and parts of $\partial\Omega_N$ it contains, picking the first hit point. Like WoS, the walk terminates inside $\partial\Omega_D^\epsilon$.

on $\partial\Omega_N$ and normal derivative $\partial u/\partial n$ on $\partial\Omega_D$. As the linear system scales quadratically in size with the number of boundary elements, BEM does not truly reduce the dimensionality of a PDE solve compared to grid-based solvers that use sparse matrices, and requires special techniques like *hierarchical matrix approximation* [Hackbusch 2015] to achieve reasonable performance. BEM does not allow for progressive evaluation, nor is output-sensitive as u and $\partial u/\partial n$ must always be determined on the entire boundary. Basic versions also ignore source terms—in this case, BEM must be coupled with a second interior solver such as FD, FEM and MFEM, inheriting shortcomings of grid-based solvers in the process [Partridge and Brebbia 2012; Costabel 1987; Coleman et al. 1991]. In contrast, our approach is progressive and output-sensitive (Section 3.2). It is also scalable to increasing geometric detail, as the underlying pointwise estimators only require a spatial hierarchy to perform queries [Sawhney et al. 2023, Section 5] and work with geometric representations beyond boundary meshes ([Sawhney and Crane 2020, Figure 2] & [Sawhney et al. 2023, Figure 20]).

The (weakly) singular yet smoothly decaying Green's function and its normal derivative (Figure 3) in Equation (2) can be problematic to integrate near the boundary with both BEM and our method (Figure 11). For BEM, specialized quadrature schemes can mitigate numerical issues due to singularities [Fairweather and Karageorghis 1998; Chen 2002, 2009]. For us, not *importance sampling* these functions leads to *local* artifacts that adversely affect estimation variance. Section 3.3 provides strategies to address this issue.

After computing u and $\partial u/\partial n$ on the boundary with BEM, Equation (2) is often evaluated using fast multipole or Barnes-Hut schemes [Greengard and Rokhlin 1987; Pfalzner and Gibbon 1997]. These acceleration strategies are often necessary due to the quadratic complexity of evaluating the BIE. We leave this optimization to future work as our current bottleneck is not BIE evaluation, but rather computing pointwise estimates on the boundary.

Sample Reuse in Rendering. Popular sample reuse schemes in rendering such as virtual point lights [Keller 1997; Dachsbacher et al. 2014], photon mapping [Jensen 1996; Hachisuka and Jensen 2009] and ReSTIR [Bitterli et al. 2020; Ouyang et al. 2021] share samples across pixels to amortize the cost of long ray-traced paths and inject global information into per-pixel radiance estimates. As a result, they are often more efficient at rendering scenes with complex geometry and lighting than brute-force path tracers.

Our approach shares similarities with VPLs and photon mapping, in that samples generated and deposited on the scene (for us $\partial\Omega$) determine the flux (*i.e.*, the solution u) over the image plane (which for us is either the entire domain Ω or a subset thereof). Unlike photon mapping however, our caching strategy does not require an additional data structure like a *kd-tree* to store samples; instead we opt for a progressive formulation that discards boundary and source samples after splatting solution and gradient estimates in Ω. Similarly to VPL methods, Monte Carlo noise is visually suppressed [Dachsbacher et al. 2014, Figure 1] from a combination of introducing correlations between estimates in Ω, and the smooth decay of the functions G and $\partial G/\partial n$ away from the boundary. VPLs are also prone to singularities, as sharing samples requires sacrificing perfect importance sampling [Dachsbacher et al. 2014, Section 5 & Figure 9]; our artifact correction schemes take inspiration from similar techniques for VPLs [Kollig and Keller 2006]. Unlike VPL methods, we do not require testing for occlusion between deposited samples and evaluation points as the BIE contains no *visibility term*.

Techniques based on *lightcuts* [Walter et al. 2005; Yuksel 2019; Lin and Yuksel 2020] can render scenes containing thousands of VPLs in real time. These methods cluster VPLs spatially in a tree, and then probabilistically select a subset of the VPLs that make the largest contribution at a given point. Akin to fast multipole and Barnes-Hut schemes, lightcuts would likely accelerate our method in domains with many evaluation points.

3 METHOD

Our method estimates the solution $u = u_{\partial\Omega} + u_\Omega$ in Equation (2) at a set of evaluation points $evalPts := \{x_k \in \Omega\}_{k=1}^K$ in a closed domain $\Omega \subset \mathbb{R}^N$ for, *e.g.*, dense visualization of the solution. To this end, we create two caches, $boundarySamples := \{z_n, \widehat{u}(z_n), \widehat{\partial u/\partial n}(z_i)\}_{i=1}^N$ and $sourceSamples := \{y_j, f(y_j)\}_{j=1}^M$, where N and M are user-specified cache sizes. The points z_i and y_j are sampled on the boundary $\partial\Omega$ and inside the domain Ω using probability densities $p^{\partial\Omega}$ and p^Ω respectively. The pointwise estimates $\widehat{u}(z_i)$ and $\widehat{\partial u/\partial n}(z_i)$ are
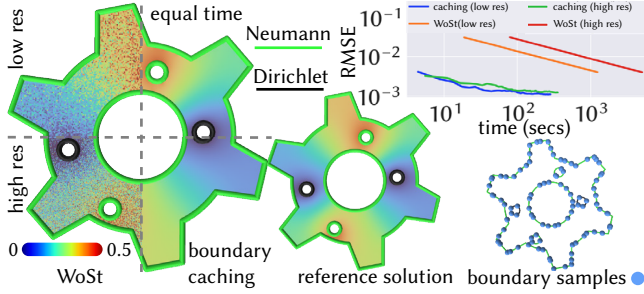
Fig. 5. We obtain far smoother results across the domain compared to directly using pointwise estimators like WoSt at equal time. Evaluating the BIE at more points inside the domain has little impact on performance, as the same boundary samples are used to determine the PDE solution.

computed using WoS(t) (Section 3.1), while $f(y_j)$ are evaluations of the known source term.

Our method then uses the two caches to form *correlated* Monte Carlo estimates of Equation (2) at all evaluation points in evalPts:

$$\widehat{u_{\partial\Omega}}(x_k) := \frac{1}{N} \sum_{i=1}^{N} \frac{\frac{\partial G}{\partial n}(x_k, z_i)\, \hat{u}(z_i) - G(x_k, z_i)\widehat{\frac{\partial u}{\partial n}}(z_i)}{p^{\partial\Omega}(z_i)}, \quad (3)$$

$$\widehat{u_{\Omega}}(x_k) := \frac{1}{M} \sum_{j=1}^{M} \frac{G(x_k, y_j)\, f(y_j)}{p^{\Omega}(y_j)}. \quad (4)$$

In Appendix A, we provide expressions for G and $\partial G/\partial n$ for the Poisson and screened Poisson equations; Algorithm 1 provides pseudocode, and Appendix B discusses the extension to open domains and double-sided boundary conditions. In the rest of this section, we detail key aspects of our approach.

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

Bias. Assuming the pointwise estimates $\hat{u}(z_i)$ and $\widehat{\partial u/\partial n}(z_i)$ are unbiased, our estimator (Equations (3) and (4)) is also unbiased via the linearity of expectation. In reality, most pointwise estimators have a small amount of controllable bias, as is well established in the WoS literature (Section 3.1). Our caching scheme does not introduce any additional bias, while greatly improving efficiency (Figure 5).

Gradient Estimation. Our method can reuse *the same* cached boundary and source samples to also form Monte Carlo estimates of the solution *gradient* at each evaluation point (Figure 7):

$$\widehat{\frac{\partial u_{\partial\Omega}}{\partial x}}(x_k) := \frac{1}{N} \sum_{i=1}^{N} \frac{\frac{\partial^2 G}{\partial x \partial n}(x_k, z_i)\, \hat{u}(z_i) - \frac{\partial G}{\partial x}(x_k, z_i)\widehat{\frac{\partial u}{\partial n}}(z_i)}{p^{\partial\Omega}(z_i)}, \quad (5)$$

$$\widehat{\frac{\partial u_{\Omega}}{\partial x}}(x_k) := \frac{1}{M} \sum_{j=1}^{M} \frac{\frac{\partial G}{\partial x}(x_k, y_j)\, f(y_j)}{p^{\Omega}(y_j)}. \quad (6)$$

We provide expressions for the derivatives of G in Appendix A.

ALGORITHM 1: A cache-based strategy to evaluate the BIE at points inside a closed user-defined boundary ∂R (Section 3.2)

```
 1: struct BoundarySample
 2:     z, n_z ← NULL            ▷Sample location & unit outward normal on ∂R
 3:     û, ∂u/∂n ← 0             ▷Estimates for solution & normal derivative
 4: struct EvaluationPoint
 5:     x ← NULL                 ▷Location for evaluating BIE (Equation (2))
 6:     û_∂Ω^sum, û_Ω^sum ← 0     ▷Running sums for solution evaluation
 7:     N, M ← 0                 ▷boundary & source sample count
 8:     function GetSolution
 9:         return û_∂Ω^sum / N + û_Ω^sum / M
10:     end function
```

Input: A set of evaluation points evalPts, cache sizes N & M for boundary and source samples, number of walks nWalks to take for pointwise estimation, and densities $p^{\partial R}$ & p^R for sample generation.

Output: An updated solution estimate for each evaluation point.

```
11: function UpdateSolution(evalPts, N, M, nWalks, p^∂R, p^R)
12:     ▷Generate N boundary samples and estimate u & ∂u/∂n for each
13:     boundarySamples ← BoundarySample[N]
14:     parallel for b in boundarySamples do
15:         b.z, b.n_z ~ p^∂R
16:         b.û, b.∂u/∂n ← WalkOnStars(b.z, b.n_z, nWalks)    ▷Sec. 3.1
17:     ▷Splat contributions from boundary samples to evalPts
18:     for b in boundarySamples do
19:         z, n_z ← b.z, b.n_z
20:         parallel for e in evalPts do
21:             e.û_∂Ω^sum += (∂G/∂n(e.x, z)b.û − G(e.x, z)b.∂u/∂n) / p^∂R(z)
22:             e.N += 1
23:     ▷Generate M source samples & splat their contributions to evalPts
24:     for j in Range(M) do
25:         y ~ p^R
26:         parallel for e in evalPts do
27:             e.û_Ω^sum += G(e.x, y) f(y) / p^R(y)
28:             e.M += 1
29: end function
```

3.1 The WoS & WoSt Estimators

We use the WoS and WoSt algorithms to compute pointwise estimates at each boundary sample z_i. WoS [Muller 1956] is a Monte Carlo algorithm that recursively evaluates the *mean-value property* of harmonic functions to solve elliptic PDEs in a bounded domain Ω with Dirichlet boundary conditions. It does so by repeatedly jumping to a random point on the *largest* sphere centered at the current random walk location (Figure 4, *left*). WoSt [Sawhney et al. 2023] generalizes WoS to solve problems with mixed Dirichlet and Neumann boundary conditions: instead of spheres wholly-contained in Ω, WoSt uses *star-shaped regions* formed by intersecting spheres with parts of the Neumann boundary $\partial\Omega_N$ (Figure 4, *right*). Both algorithms terminate walks when they enter the *epsilon shell* $\partial\Omega_D^\varepsilon$ around the Dirichlet boundary (inset), where
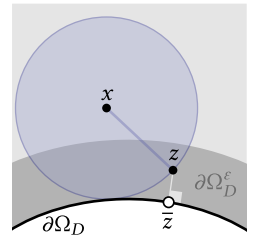

Fig. 6. Inside $\partial\Omega_D^\varepsilon$, WoS(t) uses the known Dirichlet data g from the closest projected point on $\partial\Omega_D$.
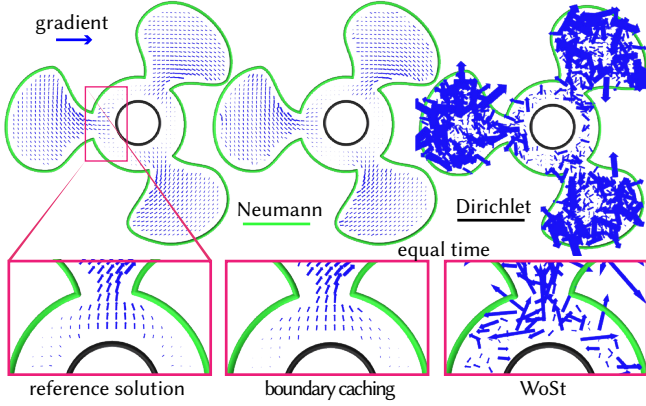
Fig. 7. Our gradients have considerably less noise compared to pointwise estimates as they use known values for $\partial u/\partial n$ on the Neumann boundary to evaluate Equation (5). In contrast, WoSt gradients become noisier away from the Dirichlet boundary as estimation requires longer random walks.

they use the known boundary value g at the nearest boundary point. This introduces a small bias into the solution estimate $\widehat{u}$, which diminishes at the rate $O(1/\log \varepsilon)$ [Binder and Braverman 2012]; we set ε to be $0.001\times$ the diagonal length of the scene.

Both WoS and WoSt replace the free-space Green's function G in Equation (2) with the Green's function G^B for a ball $B(x)$ centered at x—this choice simplifies the BIE and facilitates recursive estimation as discussed in Sawhney et al. [2023, Sections 3 & 4]. Importantly for our method, the use of G^B makes it possible to also estimate the normal derivative $\partial u/\partial n$ in Equation (2). This is done through recursive estimation of the following integral expression at any $x \in \Omega$ [Sawhney and Crane 2020, Section 3]:

$$\frac{\partial u}{\partial x}(x) = \int_{\partial B(x)} \frac{\partial^2 G^B}{\partial x \, \partial n}(x, z) \, u(z) \, \mathrm{d}z + \int_{B(x)} \frac{\partial G^B}{\partial x}(x, y) \, f(y) \, \mathrm{d}y, \quad (7)$$

where the ball $B(x)$ is selected as in WoS, and the unknown value $u(z)$ inside $\partial B(x)$ is recursively estimated using WoS for pure Dirichlet problems, and WoSt for mixed boundary-value problems. Given a unit normal n_x at x, the normal derivative is then estimated using $n_x \cdot \widehat{\partial u/\partial x}(x)$. As with $\widehat{u}$, this estimate is slightly biased due to the epsilon shell used for termination. We refer to Sawhney and Crane [2020, Equation 13 & Appendix B.1] for derivatives of G^B, and to Sawhney and Crane [2020, Section 4.1.1] and Rioux-Lavoie et al. [2022, Section 4.1] for variance reduction strategies.

3.2 Boundary Specification

When the solution needs to be evaluated within a localized region R inside the domain Ω (Figure 8), we specialize the BIE to this region by generating source samples in R and boundary samples on ∂R (Algorithm 1, *lines* 15 & 25). We use uniform densities $p^R = 1/|R|$ and $p^{\partial R} = 1/|\partial R|$ for sample generation, though we could use densities specialized to specific PDEs and their inputs to reduce variance. The solution integrates to 0 by construction outside R.

When the solution needs to be evaluated within the entire domain Ω (*i.e.*, $R := \Omega$), we incorporate the known boundary data $\partial u/\partial n = h$
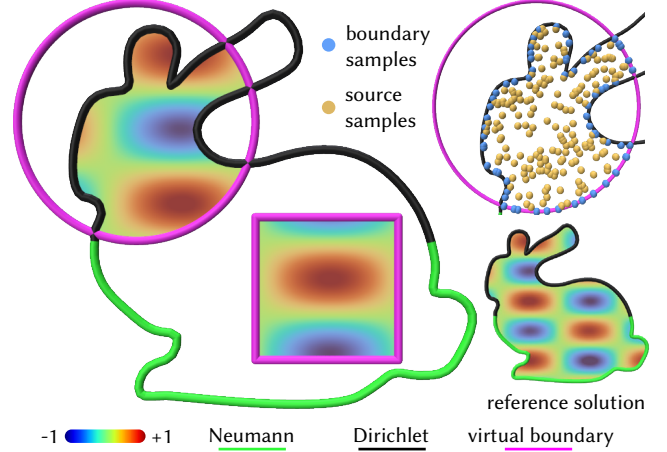

Fig. 8. Unlike BEM which must always perform a global solve involving the entire boundary, we can focus computation on a region of interest by caching points only on the boundary of smaller subdomains.
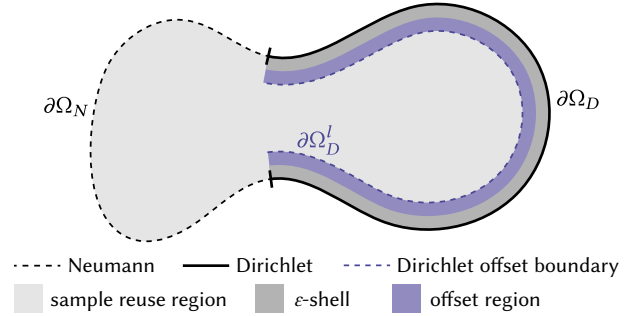

Fig. 9. To solve the BIE inside Ω, we generate samples on the Neumann boundary $\partial\Omega_N$ and an offset Dirichlet boundary $\partial\Omega_D^l$ where $l > \varepsilon$.

on $\partial\Omega_N$ directly into our sample estimates, rather than estimating it from scratch. Unfortunately, estimating $\partial u/\partial n$ on the Dirichlet boundary $\partial\Omega_D$ is challenging, as WoS uses a ball with a non-zero radius to estimate the solution gradient (Equation (7)). Instead of dealing with $\partial\Omega_D$ directly, we define a closed region bounded by the Neumann boundary $\partial\Omega_N$ and an offset Dirichlet boundary $\partial\Omega_D^l$ where $l > \varepsilon$ (Figure 9). We then generate boundary samples on $\partial\Omega_D^l$, and estimate both u and $\partial u/\partial n$. Moreover, for any evaluation point that is within a distance l to $\partial\Omega_D$, we use WoS(t) to compute the solution there, as random walks typically terminate quickly when started close to the Dirichlet boundary.

There are two considerations involved in choosing an offset l: the minimal feature size of the domain Ω, and the amount of bias in the estimate for $\partial u/\partial n$ on $\partial\Omega_D^l$ based on its proximity to the epsilon shell $\partial\Omega_D^\varepsilon$. We do not need to define sample reuse regions in the vicinity of thin features—instead, we can create multiple disconnected regions inside Ω where boundary and source samples are cached, while using WoSt to estimate the solution pointwise elsewhere. We set l to $5 \times \varepsilon$ in our results (Figure 17, *top* shows an ablation) to mitigate
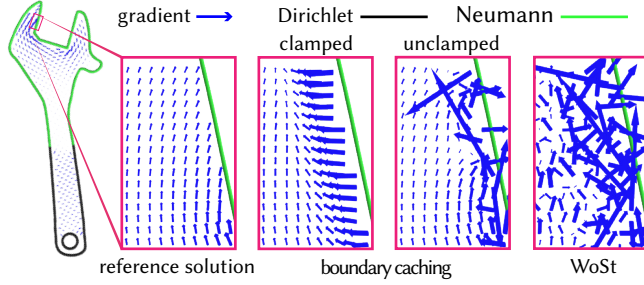
Fig. 10. Naïvely clamping singular functions in our gradient estimates suppresses noise near the boundary, but produces biased results.
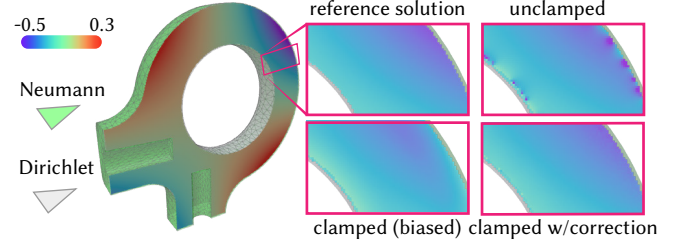

Fig. 11. Our clamping strategy (*bottom right*) effectively suppresses local artifacts from singular functions near the boundary without bias (*top right*).

bias in the estimates for $\partial u/\partial n$ on $\partial \Omega_D^I$, and leave principled and unbiased estimation of this quantity directly on $\partial \Omega_D$ to future work.

3.3 Singularities

Though the free-space Green's function and its derivatives decay smoothly, they are singular at the point they are centered on (Figure 3). Therefore, just as in BEM, our basic method can suffer from local artifacts—especially near the boundary (Figure 11)—as it uses uniformly distributed source and boundary samples to evaluate Equations (3)-(6). In contrast, such artifacts do not arise in WoS(t) as the corresponding functions for a ball are importance sampled, *i.e.*, $p^B \propto G^B$ and $p^{\partial B} \propto \partial G^B/\partial n$ [Sawhney et al. 2023, Section 4].

Similar artifacts are often suppressed in virtual point light methods by clamping the singular geometry term in the rendering equation [Dachsbacher et al. 2014, Section 5 & Figure 9]. However, clamping can introduce noticeable bias in the solution estimate with both VPLs and our method (Figure 11). Likewise, clamping derivative norms in Equations (5)-(6) leads to smoothly-varying but biased gradients near the boundary (Figure 10). Below, we extend Kollig and Keller [2006]'s bias correction strategy for VPLs to the function $\partial G/\partial n$; in Appendix C we do the same for G, and leave the extension to gradient estimators to future work.

Removing Bias from Clamping. The function $\partial G/\partial n(x, z)$ is large when the points x and z are close. To mitigate artifacts that arise from not importance sampling $\partial G/\partial n$ for z, we rewrite the first term in Equation (2) over a user-defined region R as follows:

$$\int_{\partial R} \left.\frac{\partial G}{\partial n}\right|_c (x, z) u(z)\, \mathrm{d}z \;+\; \int_{\partial R} \left[\frac{\partial G}{\partial n}(x, z) - \left.\frac{\partial G}{\partial n}\right|_c (x, z)\right] u(z)\, \mathrm{d}z,$$
(8)

where c is a positive user-specified bound, and

$$\partial G/\partial n|_c \equiv \max(-c,\ \min(c,\ \partial G/\partial n)).$$
(9)

As before, we use uniformly distributed boundary samples on ∂R to estimate the first integral in Equation (8) at any evaluation point $x \in R$. The bound c is set based on the scale of the scene (Figure 17, *bottom* shows an ablation), though strategies for automatically setting this parameter can likely be adapted from the VPL literature [Kollig and Keller 2006, Section 2.3].

To estimate the second integral, we importance-sample new boundary samples on ∂R using the probability density $p^{\partial R} = \partial G/\partial n$. For Poisson-like equations, this density defines the *signed solid angle*

over the boundary (Equation (12)), and can be sampled exactly by intersecting a ray (with a uniformly sampled direction on the unit sphere) with ∂R [Pharr et al. 2016, Section 5.5.3]. We use WoSt to estimate the unknown solution value $u(z)$ at intersection points (which might be more than one if ∂R is nonconvex; see Sawhney et al. [2023, Section 4.3]). However, we only need to estimate u when $\partial G/\partial n|_c (x, z) \neq \partial G/\partial n(x, z)$, *i.e.*, when the evaluation point x is in close proximity to an intersection point z. This is typically not the case for most evaluation points inside R, as $\partial G/\partial n$ falls off quickly (Figure 3). As a result, we can estimate Equation (8) in both an artifact- and bias-free manner by running a small number of random walks for the second integral; Figure 11 shows improvements provided by this approach.

4 IMPLEMENTATION AND EVALUATION

Setup. We encode a problem instance by a description of the domain boundary $\partial \Omega$ and the functions f, g and h in Equation (1). Unlike grid-based solvers such as FEM and BEM, these functions are not discretized or approximated in a finite basis, and are instead provided via arbitrary callback routines that return a value for any query point $x \in \Omega$. We use boundaries represented as polygonal meshes in our experiments, and a CPU-based axis aligned *bounding volume hierarchy (BVH)* to perform ray intersection and closest point queries needed by WoS(t) [Sawhney et al. 2023, Section 5]. Relative to finite element mesh generation, a BVH uses little memory and is typically a lot faster to build even for detailed models [Sawhney and Crane 2020; Sawhney et al. 2022, 2023].

Sampling. We use a discrete *cumulative density function (CDF)* table [Pharr et al. 2016, Section 13.3] with stratified random numbers to generate boundary samples over elements (*e.g.*, triangles) of a polygonal mesh. Samples are drawn uniformly on elements selected in proportion to their surface area. Faster sample generation is possible with an alias table [Walker 1974, 1977]. Source samples are likewise generated uniformly inside Ω or a user-defined region R with stratified sampling [Pharr et al. 2016, Section 13.8]. We use this sampling setup for all figures in the paper except Figure 1, where we find that results improve significantly if the boundary samples in Equations (3) and (5) are weighted by the area of their associated Voronoi cell. Similar area weighting strategies have proven effective in surface reconstruction applications [Barill et al. 2018, Figure 5], and yield a consistent Monte Carlo estimator that provides provably better convergence [Guo and Eisemann 2021].
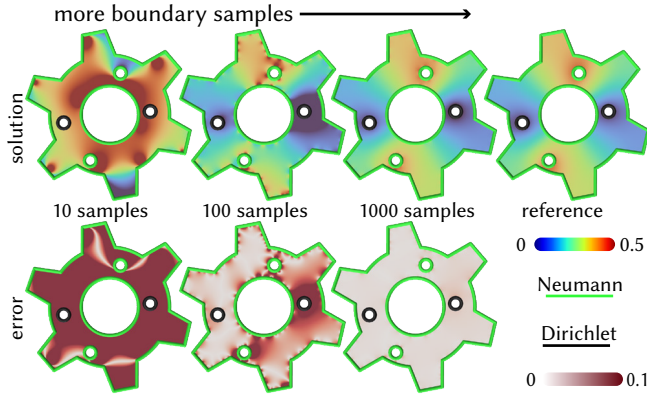
Fig. 12. Akin to traditional PDE solvers, our method demonstrates a global error in the solution profile that vanishes with more boundary samples.

Fig. 13. Compared to WoS, our caching strategy provides a noticeably smoother harmonic interpolation of texture coordinates in a deformed cage.

Convergence. In Figures 5 and 16, we show that even in 2D our method provides up to an order of magnitude error reduction at equal time compared to WoSt for mixed boundary value problems. The reason is twofold: first, run-time efficiency improves as we do not perform independent random walks for interior evaluation points. Second, boundary samples inject global information about the solution into interior estimates. The overall result is that our solver reduces the dimensionality of a PDE solve—in contrast, BEM trades off discretizing the domain with having to solve a much denser linear system.

Similarly to traditional PDE solvers requiring global solves, error in the interior now depends on the number of boundary and source samples used. Figures 12 and 16 show that error vanishes with more samples, though we note that Equations (3)-(6) provide unbiased estimates even with just a single sample. We also observe lower interior error with more accurate estimates for u and $\partial u/\partial n$ on the boundary. As derivative estimates are typically noisier than estimates of the solution [Sawhney and Crane 2020, Figure 15], we perform more random walks for the estimate $\widehat{\partial u/\partial n}$ on an offset Dirichlet boundary $\partial \Omega_D^I$ (Section 3.2) compared to $\widehat{u}$ on $\partial \Omega_N$. As a default, we take 10× more walks for boundary samples on $\partial \Omega_D^I$ than on $\partial \Omega_N$. In general, the overhead is small as walks starting close to the Dirichlet boundary are usually much shorter.

For pure Dirichlet problems, we found empirically that the bidirectional WoS approach by Qi et al. [2022] can be more effective at equal time than our more general caching strategy (Figure 14)—by specializing to Dirichlet conditions, this approach avoids the need to estimate $\partial u/\partial n$ on the boundary or evaluate $\partial G/\partial n$ during BIE estimation, which leads to lower variance.

Test Problems. Given the spatial smoothness of solutions to elliptic problems, a key strength of our method is its ability to suppress the salt-and-pepper noise characteristic of independent Monte Carlo estimates. In Figure 13, we solve a Laplace equation with pure Dirichlet boundary conditions to interpolate texture coordinates in a deformed cage. Our correlated sample estimates provide noticeably smoother results, suitable for visualization. We observe similar behavior with our gradient estimates in Figure 1, where we solve for
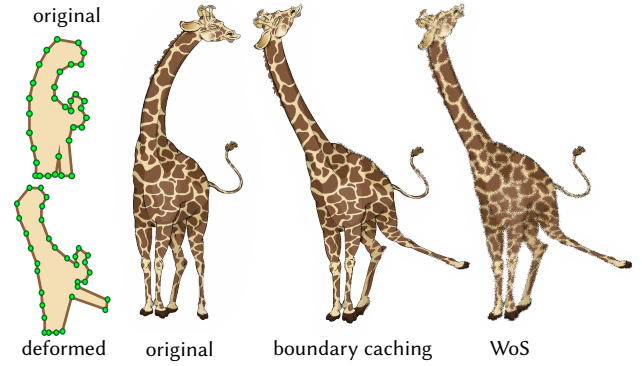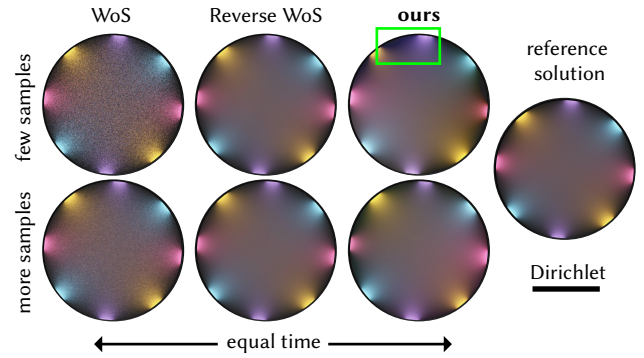

Fig. 14. For Dirichlet problems, our method often has higher variance than Qi et al. [2022]'s specialized approach for these boundary conditions.

streamlines of a potential flow with Dirichlet boundary conditions of -1 and 1 at the front and back of the tunnel respectively, and 0 Neumann conditions elsewhere. Similarly to Sawhney and Crane [2020, Section 5.2.6], we draw streamlines along the estimated gradient direction by numerically integrating an ordinary differential equation using Huen's method. Steamlines start at a collection of random seed points in the domain, and can be regenerated cheaply once estimates for u and $\partial u/\partial n$ have been computed on the boundary—we do not require a high-quality simulation mesh for this task.

In Figure 15, we demonstrate the ability of our solver to resolve detailed boundary conditions and source terms without the need for any volumetric mesh generation. In particular, we model how effectively a spacesuit regulates an astronaut's body temperature. We use a *ray-traced* solar radiation simulation for Dirichlet conditions; Neumann conditions model outflow; the source term provides the ambient body temperature. Compared to pointwise estimators, our solver provides low-variance results on a high-resolution slice plane at equal time. Moreover, efficiency improves drastically when evaluating inside a dense 3D grid since we can reuse cached samples. In this case however, evaluating the BIE at all grid points is more expensive than estimating u and $\partial u/\partial n$ on the boundary.
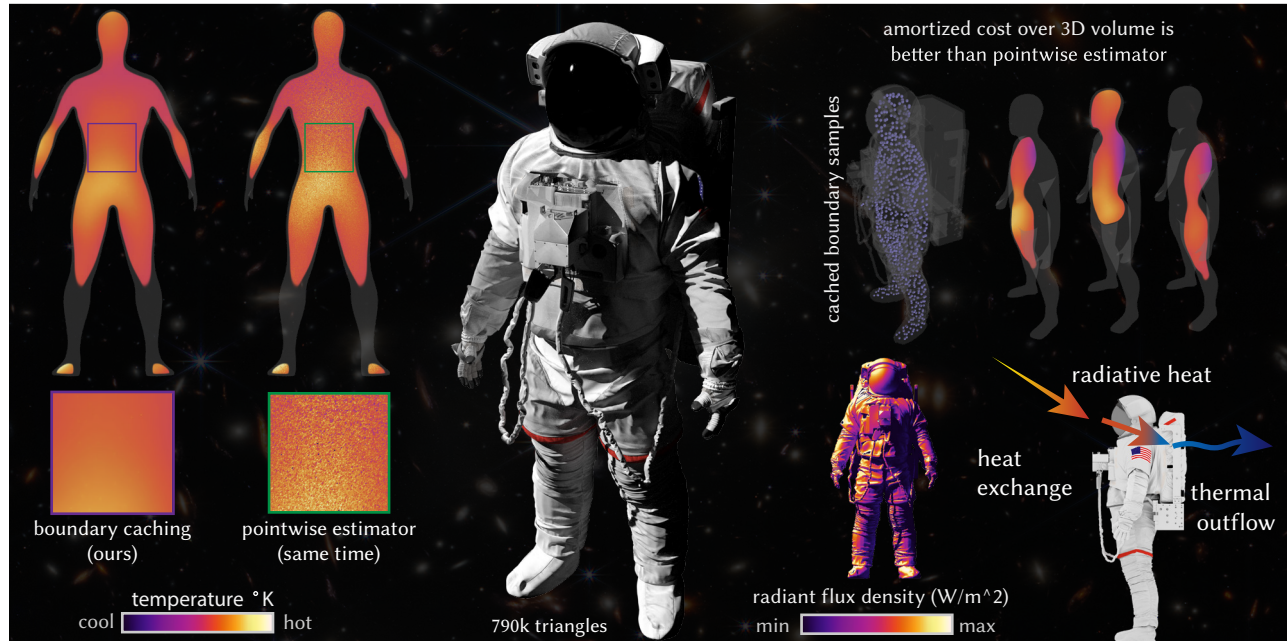
Fig. 15. Regulating the body temperature of astronauts during extra-vehicular activities is a crucial function of spacesuits. Inspired by this application, we build a simplified physical model involving heat exchange (via Dirichlet conditions) and outflow (via Neumann conditions) on a spacesuit with an interior source term to model an astronaut's ambient body temperature.

5 CONCLUSION AND FUTURE WORK

We presented a simple caching strategy for an emerging class of PDE solvers based on grid-free Monte Carlo methods. Our method greatly improves the efficiency of densely evaluating a PDE solution and its gradient inside a domain (or a subset thereof), by performing random walks only on the boundary. It also significantly suppresses Monte Carlo noise without adversely affecting scalability, progressivity, and output sensitivity of the underlying pointwise estimators.

Looking ahead, the ability of our solver to locally evaluate solutions inside arbitrary bounding regions opens up the possibility of developing domain decomposition strategies [Chan and Mathew 1994] in the Monte Carlo framework that more effectively handle domains with, *e.g.*, thin features where pointwise estimators struggle due to long random walk lengths (see Sawhney et al. [2023, Section 7]). Likewise, our method offers opportunities for improving performance of the pointwise estimators it uses by terminating walks in regions where boundary values have previously been cached.

Our method can be improved in a number of ways. In particular, faster and less noisy WoS(t) estimators will not just improve the run-time performance of our approach, they will also reduce the global error in the estimated solution inside the domain. Error will likely reduce noticeably with a principled and unbiased technique for pointwise estimation of the normal derivative of the solution on the Dirichlet boundary. Variance can be improved by adaptively generating source and boundary samples from probability densities specialized to specific PDEs and their inputs. Moreover, though sharing samples necessitates a lack of importance sampling, taking further inspiration from VPL methods [Dachsbacher et al. 2014,

Section 5] for removing singular artifacts will improve the quality of our results near the boundary, especially for gradient estimates. The quadratic complexity of evaluating the BIE can also be greatly accelerated in situations where estimation is needed at numerous points in the domain, by adapting and incorporating clustering techniques such as Barnes-Hut and lightcuts into our method.

Finally, though our method dramatically reduces the total number of random walks needed to solve PDEs relative to WoS(t), further investigation is needed to also reduce the large amounts of redundant computation performed across walks starting from the boundary—it is indeed wasteful to use each sphere in a walk to only estimate the solution at a single sample point at the very beginning of a walk.

REFERENCES

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 (jul 2018), 12 pages. https://doi.org/10.1145/3197517.3201337

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

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting. *ACM Trans. Graph.* 39, 4, Article 148 (aug 2020), 17 pages. https://doi.org/10.1145/3386569.3392481
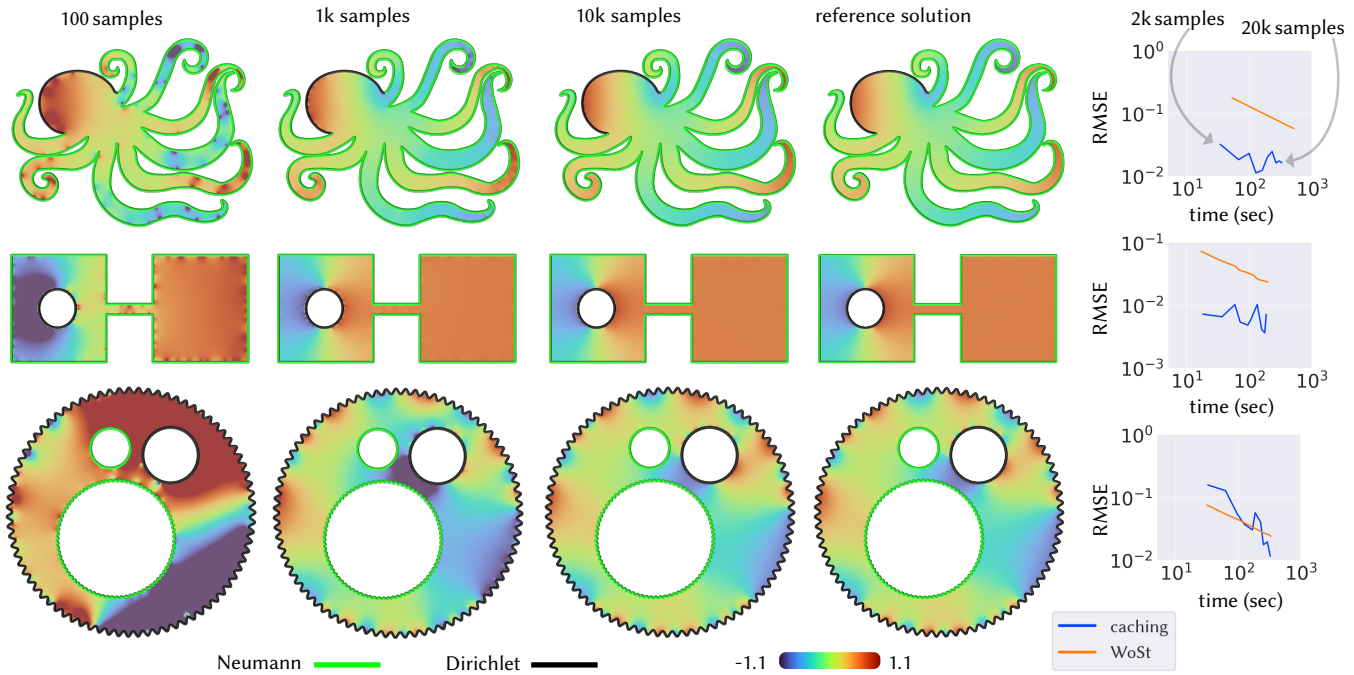
Fig. 16. Our caching strategy amortizes the cost of long walks in Neumann dominated problems (*top two rows*) and hence improves efficiency over point estimators like WoSt. However, in more Dirichlet dominated problems with higher frequency boundary conditions (*bottom row*), efficiency drops due to shorter walk lengths, not importance sampling the singular functions in the BIE (Equation (2)), and noise from estimates of $\partial u / \partial n$ on the boundary.

Tony F Chan and Tarek P Mathew. 1994. Domain decomposition algorithms. *Acta numerica* 3 (1994), 61–143.

Wen Chen. 2002. Symmetric boundary knot method. *Engineering Analysis with Boundary Elements* 26, 6 (2002), 489–494.

Wen Chen. 2009. Singular boundary method: a novel, simple, meshfree, boundary collocation numerical method. *Chinese Journal of Solid Mechanics* 30, 6 (2009), 592–599.

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.

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

Graeme Fairweather and Andreas Karageorghis. 1998. The method of fundamental solutions for elliptic boundary value problems. *Advances in Computational Mathematics* 9, 1 (1998), 69–95.

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

Jerry Jinfeng Guo and Elmar Eisemann. 2021. Geometric Sample Reweighting for Monte Carlo Integration. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 109–119.

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic Progressive Photon Mapping. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–8. https://doi.org/10.1145/1618452. 1618487

Wolfgang Hackbusch. 2015. *Hierarchical matrices: algorithms and analysis*. Vol. 49. Springer.

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

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

Alexander Keller. 1997. Instant Radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. USA, 49–56. https://doi.org/10.1145/258734.258769

Thomas Kollig and Alexander Keller. 2006. Illumination in the Presence of Weak Singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*. Springer Berlin Heidelberg, Berlin, Heidelberg, 245–257.

Daqi Lin and Cem Yuksel. 2020. Real-Time Stochastic Lightcuts. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 1, Article 5 (may 2020), 18 pages. https://doi.org/10.1145/ 3384543

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

Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. 2021. ReSTIR GI: Path Resampling for Real-Time Path Tracing. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 17–29.

Paul William Partridge and Carlos Alberto Brebbia. 2012. *Dual reciprocity boundary element method*. Springer Science & Business Media.

Susanne Pfalzner and Paul Gibbon. 1997. *Many-body tree methods in physics*. Cambridge University Press.

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

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

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, 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 (aug 2023), 22 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

Benjamin Segovia, Jean-Claude Iehl, Richard Mitanchey, and Bernard Péroche. 2006. Bidirectional instant radiosity.. In *Rendering Techniques*. 389–397.

Alastair J Walker. 1974. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters* 8, 10 (1974), 127–128.
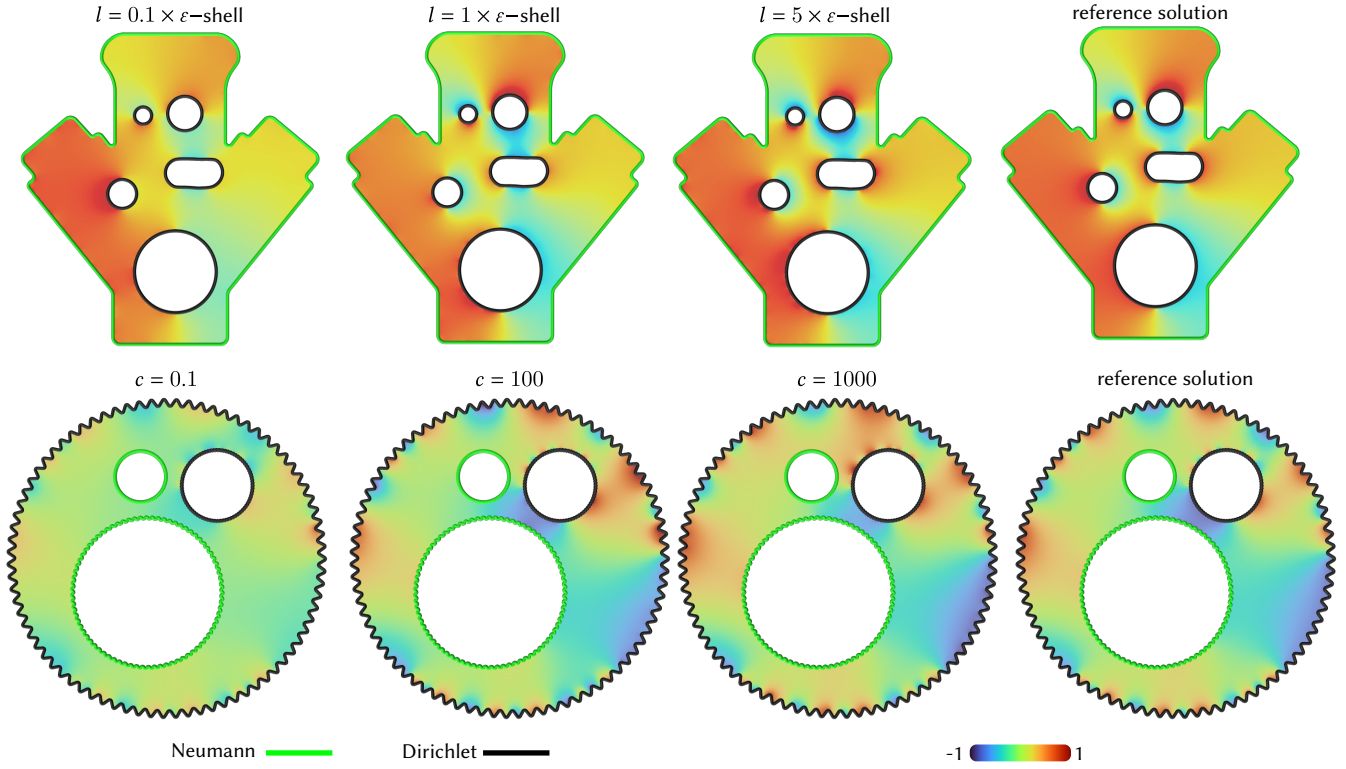
Fig. 17. *Top:* Setting the l offset parameter to $0.1 \times \varepsilon$ effectively sets each Dirichlet boundary sample's $\partial u / \partial n$ estimate to zero, as balls centered at each sample point are contained entirely inside the epislon shell—this biases the solution estimate inside the domain. Bias diminishes with increasing offset values. *Bottom:* Smaller values of the bound c for $\partial G / \partial n$ in Equation (8) suppress singular artifacts near the boundary but biases interior estimates without our correction strategy, shown here on a model scaled to fit inside a unit sphere.

Alastair J Walker. 1977. An Efficient Method for Generating Discrete Random Variables with General Distributions. *ACM Trans. Math. Softw.* 3, 3 (sep 1977), 253–256. https://doi.org/10.1145/355744.355749

Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P Greenberg. 2005. Lightcuts: A Scalable Approach to Illumination. *ACM Trans. Graph.* 24, 3 (jul 2005), 1098–1107. https://doi.org/10.1145/1073204.1073318

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).

Cem Yuksel. 2019. Stochastic Lightcuts. In *Proceedings of the Conference on High-Performance Graphics* (Strasbourg, France) *(HPG '19)*. Eurographics Association, Goslar, DEU, 27–32. https://doi.org/10.2312/hpg.20191192

A GREEN'S FUNCTIONS AND THEIR DERIVATIVES

Here we provide the free-space Green's functions and their derivatives our caching strategy uses to solve Poisson and screened Poisson equations.

A.1 Poisson Equation

The free-space Green's functions in 2D and 3D for two points x and y are given by

$$G_{2D}(x,y) = \frac{\log(r)}{2\pi}, \qquad G_{3D}(x,y) = \frac{1}{4\pi r}, \qquad (10)$$

where $r \equiv |y - x|$. Their gradients with respect to x are

$$\frac{\partial G_{2D}(x,y)}{\partial x} = \frac{x-y}{2\pi r^2}, \qquad \frac{\partial G_{3D}(x,y)}{\partial x} = \frac{x-y}{4\pi r^3}. \qquad (11)$$

The normal derivatives with respect to y are

$$\frac{\partial G_{2D}(x,y)}{\partial n_y} = \frac{n_y \cdot (y-x)}{2\pi r^2}, \qquad \frac{\partial G_{3D}(x,y)}{\partial n_y} = \frac{n_y \cdot (y-x)}{4\pi r^3}, \qquad (12)$$

where n_y is a unit normal direction associated with y. Expressions for $\partial^2 G / \partial x \partial n_y$ are given by

$$\frac{\partial^2 G_{2D}(x,y)}{\partial x \, \partial n_y} = 2\frac{n_y \cdot (y-x)}{2\pi r^4}(y-x) - \frac{n_y}{2\pi r^2}, \qquad (13)$$

$$\frac{\partial^2 G_{3D}(x,y)}{\partial x \, \partial n_y} = 3\frac{n_y \cdot (y-x)}{4\pi r^5}(y-x) - \frac{n_y}{4\pi r^3}. \qquad (14)$$

A.2 Screened Poisson Equation

We denote by K_n (for $n = 0, 1, \ldots$) the modified Bessel functions of the second kind. For a screened Poisson equation with a screening coefficient $\sigma \in \mathbb{R}_{\geq 0}$, the 2D and 3D free-space Green's functions are given by

$$G_{2D}^\sigma(x,y) = \frac{K_0(r\sqrt{\sigma})}{2\pi}, \qquad G_{3D}^\sigma(x,y) = \frac{e^{-r\sqrt{\sigma}}}{4\pi r}. \qquad (15)$$

Their gradients with respect to x are

$$\frac{\partial G_{2D}^{\sigma}(x, y)}{\partial x} = Q_{2D}^{\sigma}(x, y) \, \frac{\partial G_{2D}(x, y)}{\partial x}, \tag{16}$$

$$\frac{\partial G_{3D}^{\sigma}(x, y)}{\partial x} = Q_{3D}^{\sigma}(x, y) \, \frac{\partial G_{3D}(x, y)}{\partial x}, \tag{17}$$

where $\partial G / \partial x$ are the corresponding free-space gradients for the Poisson equation, and

$$Q_{2D}^{\sigma}(x, y) \equiv K_1(r\sqrt{\sigma}) \, r\sqrt{\sigma}, \tag{18}$$

$$Q_{3D}^{\sigma}(x, y) \equiv e^{-r\sqrt{\sigma}} \left(r\sqrt{\sigma} + 1 \right). \tag{19}$$

The normal derivatives with respect to y are

$$\frac{\partial G_{2D}^{\sigma}(x, y)}{\partial n_y} = Q_{2D}^{\sigma}(x, y) \, \frac{\partial G_{2D}(x, y)}{\partial n_y}, \tag{20}$$

$$\frac{\partial G_{3D}^{\sigma}(x, y)}{\partial n_y} = Q_{3D}^{\sigma}(x, y) \, \frac{\partial G_{3D}(x, y)}{\partial n_y}. \tag{21}$$

Applying the product rule to the expressions above yields $\partial^2 G^{\sigma} / \partial x \partial n_y$.

B OPEN DOMAINS AND DOUBLE-SIDED BOUNDARY CONDITIONS

For a domain $\Omega \subset \mathbb{R}^N$ with open boundaries and double-sided boundary conditions, the BIE in Equation (2) for can be generalized to [Costabel 1987]

$$\begin{aligned} u(x) = & \int_{\partial\Omega} \frac{\partial G}{\partial n^+}(x, z) \, \left[u^+(z) - u^-(z) \right] \\ & - G(x, z) \left[\frac{\partial u^+}{\partial n^+}(z) - \frac{\partial u^-}{\partial n^-}(z) \right] \, dz \\ & + \int_{\Omega} G(x, y) \, f(y) \, dy, \end{aligned} \tag{22}$$

for any point $x \in \Omega$, where n^+ and n^- denote unit outward and inward facing normals on $\partial\Omega$ respectively, and u^+ and u^- represent corresponding solution values on either side of $\partial\Omega$. From a sample reuse perspective, we generate uniformly distributed source samples inside Ω as usual (if Ω is unbounded, we create a bounding box around it), but now use two separate sets of boundary samples associated with the two normal directions n^+ and n^- on $\partial\Omega$ respectively. As discussed in Sawhney et al. [2023, Appendix B], we can use WoSt to estimate u^+ and u^- by launching random walks on either side of the Neumann boundary $\partial\Omega_N$. Similar to Section 3.2, we generate boundary samples on two offset Dirichlet boundaries $\partial\Omega_D^l$ and $\partial\Omega_D^{-l}$ to estimate $\partial u^+ / \partial n^+$ and $\partial u^- / \partial n^-$ with WoS respectively. With this setup, we can then compute a solution estimate for any evaluation point $x \in \Omega$ by using all source and boundary samples, except for points that are within a distance l to $\partial\Omega_D$ where we use WoSt to compute pointwise solution estimates.

C MITIGATING SINGULAR ARTIFACTS FROM THE GREEN'S FUNCTION

We follow a similar recipe to Equation (8) to mitigate localized artifacts that arise from not importance sampling the Green's function in the BIE's source integral. In particular, if c is a positive user-specified bound and $G_c \equiv \min(-c, \max(c, G))$, then we can rewrite

the source integral over a region R as follows:

$$\int_R G_c^B(x, y) \, f(y) \, dy \; + \; \int_R \left[G^B(x, y) - G_c^B(x, y) \right] f(y) \, dy, \tag{23}$$

where rather than the free-space Green's function G, we now use the Green's function of a ball G^B (see Sawhney et al. [2023, Appendix A]). The reason we use G^B instead of G is because it can be importance-sampled for y [Sawhney et al. 2022, Section 1.4, Supplemental]. It also shares the same normal derivative as its free-space counterpart, which allows G^B to be used in the BIE's boundary integral as well. We choose the ball B to be centered at the evaluation point x such that it contains the region R inside it. The first integral in Equation (23) is then estimated as usual by using uniformly distributed source samples in R. The second integral is estimated by sampling G^B for y and setting $f = 0$ if $y \notin R$.