Differential Walk on Spheres

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

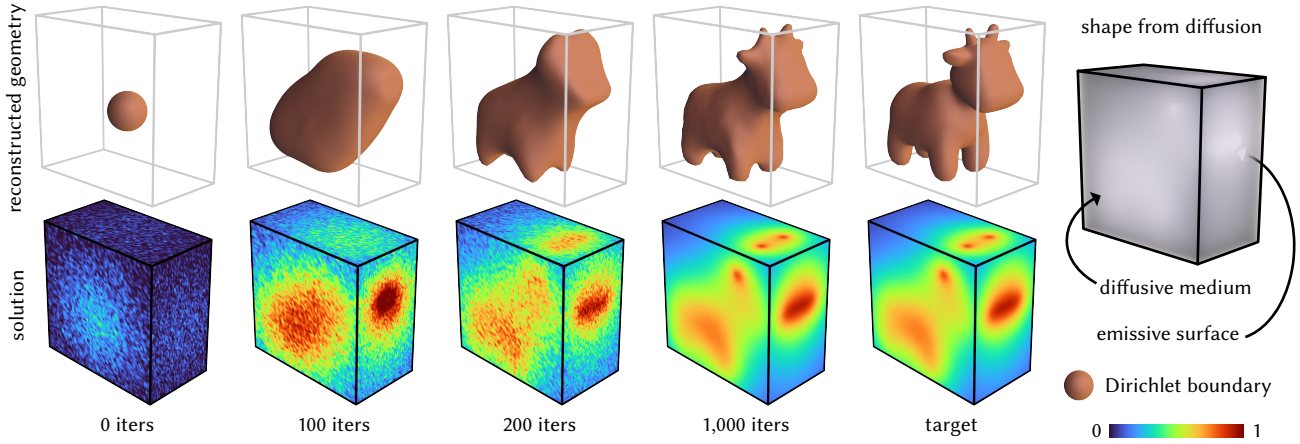


Figure 1. For a given boundary value problem, our differential walk on spheres algorithm makes it possible to differentiate solution values with respect to problem parameters. Here we consider an inverse problem where we recover the shape of an emissive object from its observed diffusion profile on the boundary of a box, via gradient-based optimization. Unlike conventional mesh- or grid-based approaches, we can evaluate derivatives at points of interest without needing to compute a global solution (here, only at the observed points).

We introduce a Monte Carlo method for computing derivatives of the solution to a partial differential equation (PDE) with respect to problem parameters (such as domain geometry or boundary conditions). Derivatives can be evaluated at arbitrary points, without performing a global solve or constructing a volumetric grid or mesh. The method is hence well suited to inverse problems with complex geometry, such as PDE-constrained shape optimization. Like other *walk on spheres (WoS)* algorithms, our method is trivial to parallelize, and is agnostic to boundary representation (meshes, splines, implicit surfaces, *etc.*), supporting large topological changes. We focus in particular on screened Poisson equations, which model diverse problems from scientific and geometric computing. As in differentiable rendering, we jointly estimate derivatives with respect to all parameters—hence, cost does not grow significantly with parameter count. In practice, even noisy derivative estimates exhibit fast, stable convergence for stochastic gradient-based optimization, as we show through examples from thermal design, shape from diffusion, and computer graphics.

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, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA; Ioannis Gkioulekas, igkioule@cs.cmu.edu, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA.

ACM 0730-0301/2024/12-ART174
https://doi.org/10.1145/3687913

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Rendering**.

Additional Key Words and Phrases: Walk on spheres, differentiable simulation, shape optimization

ACM Reference Format:
Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2024. Differential Walk on Spheres. *ACM Trans. Graph.* 43, 6, Article 174 (December 2024), 18 pages. https://doi.org/10.1145/3687913

1 INTRODUCTION

Which shape best explains observed physical behavior? How can one design shapes that maximize (or minimize) a target physical quantity? Such *inverse problems* are fundamental to numerous challenges in science and engineering. For instance, one might need to assess damage to an airplane wing using indirect thermal measurements [Zalameda and Parker 2014], or infer the shape of a tumor through deep layers of tissue [Arridge 1999]. Likewise, one might seek to design circuit geometry that maximizes dissipation of heat [Zhan et al. 2008], airfoils that generate prescribed lift [Hicks and Henne 1977], or lightweight structures that withstand significant load [Allaire et al. 2014]. To solve such problems, one must be able to efficiently and accurately differentiate solutions to partial differential equations (PDEs) with respect to the shape of the domain, or its boundary conditions [Hadamard 1908; Céa et al. 1973]. However, for problems with complex geometry, even just solving such PDEs on a fixed domain can be daunting for traditional methods—making many important inverse problems unapproachable.

The *walk on spheres (WoS)* method [Muller 1956; Sawhney and Crane 2020] and its recent extensions [Sawhney et al. 2022, 2023; Miller et al. 2024] provide *grid-free* alternatives to traditional PDE solvers that entirely bypass the need for volumetric mesh generation or global solves. Compared to finite element or finite difference techniques, these Monte Carlo-based methods enable *pointwise* evaluation of the solution to linear elliptic PDEs such as the Poisson equation, through simulation of independent random walks (Figure 2, *left*). As explored by Sawhney and Crane [2020] and in followup work [Sawhney et al. 2022, 2023; Miller et al. 2024], this approach offers a degree of geometric robustness and flexibility that has previously eluded traditional solvers. Yet WoS has historically been limited to *forward* tasks (*i.e.*, solving PDEs on a given domain), rather than inverse tasks such as geometric optimization.

How can we adapt WoS to inverse problems? One idea is to apply *automatic differentiation* to existing algorithms—but as in differentiable rendering this naïve strategy is undesirable for several reasons: 1. Automatic differentiation, when naïvely applied to algorithms like WoS and Monte Carlo path tracing, computes derivatives for accelerated geometric queries which is both inefficient and impractical [Niemeyer et al. 2020]. 2. It results in exponential memory complexity and quadratic computational complexity, compared to the linear memory and computational complexity of the recursive forward algorithms [Nimier-David et al. 2020; Vicini et al. 2021]. 3. It cannot compute correct derivatives with respect to the evolving geometry that geometric optimization and inverse problems typically require [Li et al. 2018; Loubet et al. 2019; Bangaru et al. 2020; Zhang et al. 2020]. 4. Even in constrained settings where these critical limitations may not apply, automatic differentiation of Monte Carlo algorithms results in differential estimators with suboptimal statistical performance [Zeltner et al. 2021].

We instead take a more principled approach, and formulate PDEs that compute derivatives (*a.k.a. sensitivities*) with respect to a collection of parameters π. Crucially, these PDEs can again be estimated via WoS algorithms—which must now be adapted to handle nesting between the PDEs characterizing the derivatives, and the solution itself (Section 4.1). The parameters π are analogous to the *scene parameters* in differentiable rendering—for example, they might describe the vertex locations of a triangular mesh, the anchors and tangents of a cubic spline, the centers of radial basis functions defining an implicit surface, or the pose of a rigid object. Our method simultaneously estimates derivatives for all parameters using a set of nested random walks, making it feasible to optimize systems with thousands of degrees of freedom (*e.g.*, a densely-sampled mesh).

The Monte Carlo approach is also well-matched to the needs of descent-based optimization. Far from local minima, the gradient is merely a heuristic for the ideal descent direction—making exact gradient computation (as in traditional methods) overkill in most scenarios. A stochastic approach enables us to reduce the computational load in early stages of optimization by taking fewer samples, increasing sample count only as we approach a solution. In fact, as in differentiable rendering [Gkioulekas et al. 2013; Li et al. 2018; Zhang et al. 2020; Nimier-David et al. 2020; Vicini et al. 2022; Zhao et al. 2020], noisy derivatives can actually *improve* optimization quality by helping to avoid local minima (Section 6.2). Moreover, our derivatives do not suffer from numerical robustness issues that can plague traditional PDE solvers, such as exploding derivatives due to ill-conditioned elements on an evolving mesh.

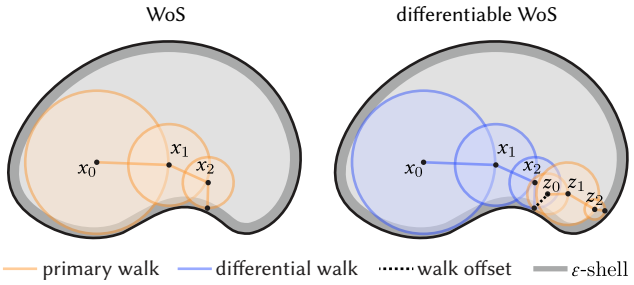


Figure 2. *Left:* Walk on spheres (Algorithm 1) recursively jumps to a random point on the largest sphere around the current walk location. The walk is terminated when it reaches the ε-shell, where the Dirichlet condition is evaluated at the closest point on the boundary. *Right:* Differential WoS (Algorithm 2) also takes a walk to the ε-shell, but additionally launches a primary walk from an offset point close to the boundary to estimate the differential boundary condition.

Scope. Although our approach applies in principle to many of the PDEs handled by current (forward) WoS algorithms (*e.g.*, with drift terms or variable coefficients), we choose to focus this paper on *screened Poisson equations*, which strike a good balance between generality and conceptual and expositional simplicity. Such equations find widespread use in geometry processing, simulation, graphics, and scientific computing. For example, they can model the steady-state temperature inside a conductive solid, the isotropic distribution of light deep inside a scattering medium, or the diffusive interpolation of boundary values in an inpainted image. We show geometric optimization examples inspired from these settings in Section 7, and choose these examples to also showcase the ability of our approach to optimize complex and varied geometric representations—triangular meshes, Bézier curves, and implicit surfaces. We provide a video visualizing these optimization examples, and an open-source implementation of our solver, on the project website.[1]

2 RELATED WORK

Optimizing the parameters of a system, such as its shape, to improve performance or meet a desired objective is a widely studied mathematical and algorithmic topic. Here we broadly review prior work on inverse design problems that involve solving PDEs, followed by Monte Carlo methods for differentiable simulation in graphics.

2.1 PDE-constrained shape optimization

Many application domains require studying how the shape of a system evolves under small changes to a PDE solution—for the transfer of heat [Zhuang et al. 2007; Feppon 2019], deformation of elastic bodies [Ameur et al. 2004; Allaire et al. 2014], flow of fluids [Treuille et al. 2003], distortion of mappings [Sharp and Crane 2018], interpolation of colors [Zhao et al. 2017], *etc.* The mathematical roots of PDE-constrained shape optimization can be traced back to the method of Hadamard [Hadamard 1908], which computes the sensitivity of

[1] https://imaging.cs.cmu.edu/differential_walk_on_spheres

a problem subject to small deformations of its boundary. With the advent of powerful computers and advanced numerical methods like the finite element method (FEM), the mathematical framework of shape optimization was further developed by authors such as Céa [Céa et al. 1973; Cea et al. 1974], wherein techniques such as implicit differentiation and the adjoint method became popular for efficiently computing derivatives of objective functions dependent on PDE solutions [Henrot and Pierre 2018]. We use these classical results, specifically for the screened Poisson equation as reviewed in Section 4, as the starting point for developing our differential Monte Carlo method in Section 5. Note that Zhao et al. [2017] and Henrot and Pierre [2018, Section 5.8] show how to express shape-functional derivatives in terms of the solution to an *adjoint boundary value problem*—we compare this formulation to ours in Appendix B.

2.2 Algorithms for shape optimization

The earliest numerical methods for shape optimization were based on evolving meshes [Pironneau 1984; Allaire and Pantz 2006], where a discretized domain serving as an initial guess is deformed under PDE constraints to better match an objective. Over the past two decades, level-set based methods [Ameur et al. 2004; Li et al. 2005; Zhuang et al. 2007; Allaire et al. 2014; Feppon 2019; Lebbe et al. 2019; Gropp et al. 2020] have since superseded purely mesh-based approaches—here a shape described implicitly as a level set function is evolved on a fixed background mesh or grid. This representation enables complex topological changes to the optimized shape (*e.g.*, merging of holes, region splitting) without needing to remesh, which is particularly desirable for optimizations where the final shape is unknown. Hybrid approaches have also been developed, allowing for both topological changes and more accurate PDE solutions via conforming mesh boundaries [Allaire et al. 2013]. More recently, *physics-informed neural networks* have emerged as alternatives to traditional PDE techniques [Raissi et al. 2019]; these methods can face difficulties in enforcing hard PDE constraints in complex geometric domains [Krishnapriyan et al. 2021], and are currently in their infancy for shape optimization tasks [Lu et al. 2021].

Grid-based methods in graphics. Computer graphics has developed specialized algorithms for shape optimization, to handle the challenging design of fluidic systems [Du et al. 2020; Li et al. 2022], rigid structures [Zhu et al. 2017; Liu et al. 2018; Whiting et al. 2012], and acoustic filters [Li et al. 2016]. These methods build on decades of shape optimization research that leverages grid-based techniques [Feppon 2019]. Though practical for their intended applications, these methods are designed for a specific geometric representation and require global solves to compute gradients during optimization.

By contrast, our method can work directly with a variety of boundary representations, such as meshes, splines, NURBS [Marschner et al. 2021], and even implicit surfaces [Sharp and Jacobson 2022; Gillespie et al. 2024] that allow for large topological changes (Section 7), without the need for any specialization—the only requirement is to be able to query the distance to the boundary. Another key benefit of our Monte Carlo approach over grid-based alternatives is *output-sensitive optimization*, *i.e.*, the ability to evaluate derivatives pointwise in regions of interest without a *global* PDE solve (Figure 11). Our method also avoids the many challenges associated with grid generation and adaptive refinement—for instance, it does not have to contend with numerical issues faced by mesh-based approaches from ill-conditioned elements. Moreover, even though derivative estimates computed with our method contain noise, they are correct in expectation. We can therefore improve their accuracy at any stage of the optimization by simply taking more samples, in place of, *e.g.*, increasing the resolution of a background grid.

2.3 Differentiable Monte Carlo simulation

Stochastic gradient optimization has had a profound impact on machine learning, enabling efficient training of complex models on large datasets without overfitting [Kingma and Ba 2015]. In recent years, it has also become popular for differentiable physics simulation [Nimier-David et al. 2019; Hu et al. 2020; Macklin 2022] and geometric optimization [Remelli et al. 2020; Shen et al. 2021, 2023] in computer graphics. For instance, it is the method of choice for differentiable Monte Carlo rendering [Gkioulekas et al. 2013; Li et al. 2018; Zhang et al. 2020; Nimier-David et al. 2020; Vicini et al. 2022; Zhao et al. 2020], where additional stochasticity in the derivative estimates of scene parameters (*e.g.*, lighting, materials and geometry) has been observed to act as a regularizer for highly underdetermined inverse rendering problems. When paired with 3D geometry representations that allow for easy topological changes, differentiable Monte Carlo rendering reliably recovers a target geometry using a random or uninformed initialization of scene parameters [Hasselgren et al. 2022; Cai et al. 2022; Mehta et al. 2022]. Our method likewise employs stochastic optimization with noisy derivative estimates for PDE-constrained shape optimization and may be paired with a range of boundary representations. Similar to differentiable rendering, cost does not increase significantly with the number of parameters, as during optimization we simultaneously estimate derivatives with respect to all parameters with a single PDE solve.

2.4 Walk on spheres

We develop our differential Monte Carlo method using the walk on spheres algorithm [Muller 1956], which has become a popular grid-free alternative to traditional PDE solvers in the graphics community due to its striking similarities with Monte Carlo ray tracing [Sawhney and Crane 2020]. In a short span of time, WoS has been generalized to solve a much broader class of linear diffusive PDEs [Nabizadeh et al. 2021; Sawhney et al. 2022, 2023; Miller et al. 2024; De Lambilly et al. 2023]; a number of variance reduction techniques have also been developed to improve estimation quality [Qi et al. 2022; Miller et al. 2023; Bakbouk and Peers 2023; Li et al. 2023], along with explorations into more advanced applications such as thermal imaging [Bati et al. 2023] and fluid simulation [Rioux-Lavoie et al. 2022; Jain et al. 2024; Sugimoto et al. 2024]. Though we focus primarily on screened Poisson equations with Dirichlet boundary conditions, our method can in principle be generalized to handle the larger set of equations WoS can solve, *e.g.*, we describe an extension to mixed Dirichlet-Robin boundary conditions in Section 5.2. Moreover, as our approach essentially boils down to coupling two random walks simulated using WoS, it should benefit directly from any future improvements to WoS estimators.

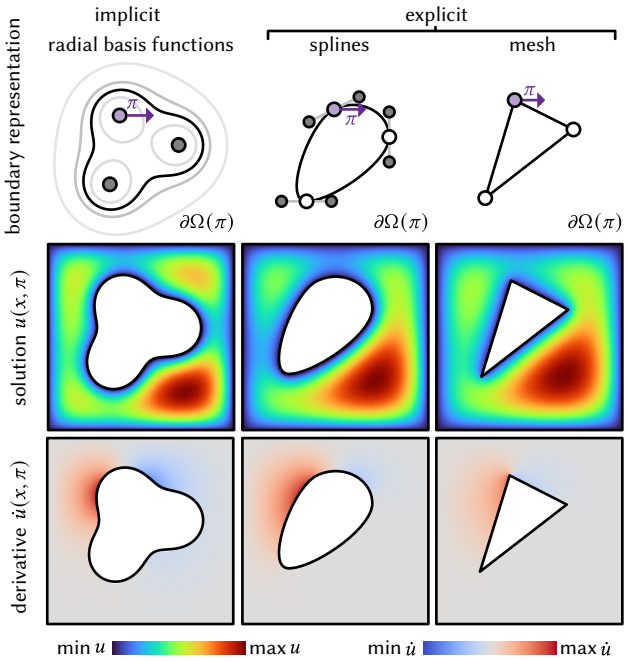


Figure 3. We can compute derivatives on a wide range of boundary representations, including those not directly handled by conventional solvers: implicit surfaces (*left*) and splines (*center*). For a solution to a Poisson equation $u(x,\pi)$ (*middle row*), here we show the derivatives $\dot{u}(x,\pi)$ (*bottom row*) with respect to positional parameters π of the boundary $\partial\Omega(\pi)$.

WoS for inverse problems. Yılmazer et al. [2022] considered a differential version of WoS for optimizing parameterized source terms and PDE coefficients on a *fixed* domain. Our emphasis is on optimizing the shape of the domain and the boundary conditions imposed on it. These differentiable simulation capabilities are complementary and useful for different types of inverse problems.

Walk on boundary alternative. The walk on boundary method [Sugimoto et al. 2023; Sabelfeld and Simonov 2013] is another Monte Carlo PDE solver that performs random walks over the boundary of a domain via ray intersection queries. Designing a differential walk on boundary method is conceptually attractive given the close resemblance of the forward process to Monte Carlo rendering; however, prior work [Sawhney et al. 2023; Miller et al. 2024] demonstrates that the practical application of walk on boundary is hindered by the lack of convergence guarantees. In particular, in the presence of either non-convex geometry or mixed boundary conditions, walk on boundary estimates can fail to converge. By contrast, walk on spheres is provably convergent and its practical application to complex domains is well demonstrated by prior works [Sawhney and Crane 2020; Sawhney et al. 2023; Miller et al. 2024].

Concurrent work. In concurrent work, Yu et al. [2024] develop a similar algorithm to ours for shape optimization constrained by the screened Poisson equation. Our work differs in a few key ways: 1. The theoretical framework we develop our algorithm on translates the problem of computing derivatives to one of solving boundary value problems (BVPs) of the same type as the PDE in consideration (Section 4). This framework allows us to easily extend our approach to more general BVPs (Sections 3.4 & 5.2) using existing estimators that generalize WoS [Sawhney et al. 2023; Miller et al. 2024], which we demonstrate in a thermal design problem that includes mixed Dirichlet-Robin boundary conditions (Section 7.3). In principle, our approach also enables extending other existing estimators for this PDE (*e.g.*, bidirectional or reverse random walks [Qi et al. 2022; Miller et al. 2024]) to the differential setting. 2. We provide an unbiased and efficient approach for calculating reverse-mode derivatives of shape functionals using *U-statistics* (Section 5.3). 3. We show more varied and challenging shape optimization applications, including optimization of complex 3D meshes, implicit geometry, and Bézier curves, to help showcase the benefits of Monte Carlo techniques for PDE-constrained shape optimization.

3 BACKGROUND

We first review concepts from surface evolution and PDE theory, as well as the walk on spheres algorithm. For in-depth discussion of differential surface evolution, we refer to Henrot and Pierre [2018].

3.1 Surface evolution

Throughout we optimize a domain $\Omega(\pi) \subset \mathbb{R}^3$ with boundary $\partial\Omega(\pi)$, whose geometry is encoded by a finite set of parameters $\pi \in \mathbb{R}^N$. (Section 7 details several possibilities.) For any π-dependent function ϕ, we denote by $\nabla\phi$ partial differentiation with respect to its domain (*i.e.*, the *spatial gradient*); and by $\dot{\phi}$ partial differentiation with respect to π (*i.e.*, the *parameter derivative*).

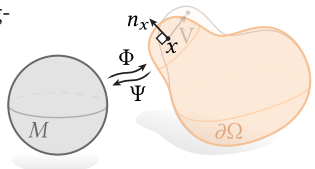
We assume that the parameterization of $\partial\Omega(\pi)$ is differentiable, so that an infinitesimal perturbation of π corresponds to an infinitesimal deformation of $\partial\Omega(\pi)$, described by a *velocity field* $\mathrm{V} : \partial\Omega(\pi) \to \mathbb{R}^3$. Since tangential motion does not change the geometry of $\partial\Omega(\pi)$, we often require only the *normal velocity* $\mathrm{V}_n := n \cdot \mathrm{V}$, where $n : \partial\Omega(\pi) \to \mathbb{R}^3$ is the outward unit normal field. Expressions for the normal velocity depend on whether we model $\partial\Omega(\pi)$ as an implicit or explicit surface (Figure 3).

3.1.1 Implicit surface. In this case, we model the boundary $\partial\Omega(\pi)$ as the zero-level set of an *implicit function* $\Gamma(\cdot,\pi) : \mathbb{R}^3 \to \mathbb{R}$ controlled by π; that is, $\partial\Omega(\pi) := \{x \in \mathbb{R}^3 : \Gamma(x,\pi) = 0\}$. Then, using the level set equation, the normal velocity at $x \in \partial\Omega(\pi)$ becomes [Stam and Schmidt 2011; Osher and Fedkiw 2005]

$$\mathrm{V}_n(x,\pi) = \frac{\dot{\Gamma}(x,\pi)}{\|\nabla\Gamma(x,\pi)\|}. \tag{1}$$

Section 7 shows optimization of an implicit surface represented as a sum of radial basis functions (harmonic monopoles).

3.1.2 Explicit surface. Here we imagine the boundary $\partial\Omega(\pi)$ is parameterized over a *reference surface* M. In particular, we view $\partial\Omega(\pi)$ as the image of a *parameterization* $\Phi(\cdot,\pi) : M \to \mathbb{R}^3$; that is $\partial\Omega(\pi) := \{\Phi(p,\pi) : p \in M\}$. Concretely, one can think of Φ as assigning coordinates to vertices of a mesh, or to control handles of a spline (Section 7). We assume that Φ is always a diffeomorphism, and hence



has a well-defined inverse $\Psi(\cdot, \pi) : \partial\Omega(\pi) \to M$. In this case, the normal velocity at $x \in \partial\Omega(\pi)$ is

$$\mathrm{V}_n(x, \pi) = n_x \cdot \dot{\Phi}(\Psi(x, \pi), \pi). \tag{2}$$

Note that the normal velocity is independent of the specific choice of parameterization [Grinfeld 2013].

3.2 Screened Poisson equation

The main PDE we consider is a *screened Poisson* boundary value problem. For simplicity we focus mainly on Dirichlet boundary conditions; Section 3.4 treats Neumann and Robin boundary conditions.

Primary boundary value problem

$$\begin{aligned} \Delta u(x) - \sigma u(x) &= \mathrm{f}(x) && \text{in } \Omega(\pi), \\ u(x) &= \mathrm{g}(x, \pi) && \text{on } \partial\Omega(\pi). \end{aligned} \tag{3}$$

Here Δ is the negative-semidefinite Laplace operator on $\mathbb{R}^3$, $\sigma \in \mathbb{R}_{\geq 0}$ is a constant *screening coefficient*, $\mathrm{f} : \Omega(\pi) \to \mathbb{R}$ is the *source term*; $\mathrm{g}(\cdot, \pi) : \partial\Omega(\pi) \to \mathbb{R}$ is the *boundary data*, which may also depend on parameters π (Section 4.1), and $u : \Omega(\pi) \to \mathbb{R}$ is the *solution* of the BVP. Although u implicitly depends on the parameters π, we typically omit this dependence to simplify notation. We refer to Equation (3) as the *primary boundary value problem*, to distinguish it from the BVP developed in Section 4.1 to express derivatives.

3.3 Walk on spheres

Walk on spheres (WoS) algorithms [Muller 1956; Sawhney and Crane 2020] can be used to estimate solutions to PDEs such as Equation (3). The basic idea is to express the solution $u(x_0)$ at a point $x_0 \in \Omega$ as an integral over a ball around x_0, in terms of the unknown function u. Applying a single-sample Monte Carlo estimate to this integral then requires evaluating $u(x_1)$, leading to a recursive estimation procedure (akin to Monte Carlo path tracing). The random walk $x_0, x_1, \ldots$ terminates when x_k gets within a small distance $\varepsilon > 0$ of the domain boundary $\partial\Omega$, where the solution is approximated using the value of g at the closest boundary point. Here we describe only the basic WoS estimator needed for our screened Poisson equation; see references in Section 2.4 for further background.

3.3.1 Notation. For any point $x \in \Omega(\pi)$, we let $\overline{x}$ be the closest point on the boundary $\partial\Omega(\pi)$, and let $\overline{\mathrm{R}}(x) := \|x - \overline{x}\|$ be the corresponding shortest distance. We use $\mathrm{B}(x, \mathrm{R})$ for the ball with center x and radius R, and $\overline{\mathrm{B}}(x) := \mathrm{B}(x, \overline{\mathrm{R}}(x))$ for the largest such ball contained in $\Omega(\pi)$. We define the *ε-shell* of $\partial\Omega(\pi)$ as the set $\partial\Omega^\varepsilon(\pi) := \{x \in \Omega(\pi) : \overline{\mathrm{R}}(x) \leq \varepsilon\}$. Finally, we denote by $\mathrm{G}, \mathrm{P} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ the Green's function and Poisson kernel (*resp.*) for the zero-Dirichlet screened Poisson equation on a ball [Sawhney et al. 2023, Appendix A]; since these functions are rotationally symmetric, we parameterize them in terms of a positive radius.

3.3.2 WoS estimator for Screened Poisson equations. We can write the recursive WoS estimator for Equation (3) at point x_0 as follows:

Walk on spheres estimator

$$\widehat{u}(x_k) := \begin{cases} \mathrm{g}(\overline{x_k}), & x_k \in \partial\Omega^\varepsilon, \\ \frac{\mathrm{P}(\overline{\mathrm{R}}(x_k))}{p^{\partial\mathrm{B}}(x_k)} \widehat{u}(x_{k+1}) + \frac{\mathrm{G}(\|z_k - x_k\|)}{p^{\mathrm{B}}(x_k)} \mathrm{f}(z_k), & \text{otherwise.} \end{cases} \tag{4}$$

The next walk point x_{k+1} and source point z_k are sampled from the sphere $\partial\overline{\mathrm{B}}(x_k)$ and ball $\overline{\mathrm{B}}(x_k)$ with uniform probability $p^{\partial\mathrm{B}}(x_k) := 1/|\partial\overline{\mathrm{B}}(x_k)|$ and $p^{\mathrm{B}}(x_k) := 1/|\overline{\mathrm{B}}(x_k)|$ (resp.).

Russian roulette. When $\sigma > 0$, contributions at each walk step are attenuated by a multiplicative factor $\alpha \in [0, 1)$ (*line 12*, Algorithm 1). We apply Russian roulette [Pharr et al. 2023, Section 2.2.4] proportionally to α to stochastically terminate walks before they reach $\partial\Omega^\varepsilon(\pi)$. Early termination improves the efficiency of the WoS estimator, without introducing additional bias. We summarize the resulting WoS procedure with Russian roulette in Algorithm 1.

3.4 Mixed Dirichlet-Robin boundary conditions

We can generalize our theory and algorithms to screened Poisson equations with mixed Dirichlet and Robin boundary conditions. Suppose we partition the domain boundary $\partial\Omega(\pi)$ into a *Robin boundary* $\partial\Omega_\mathrm{R}$ and a *Dirichlet boundary* $\partial\Omega_\mathrm{D}(\pi)$. Our BVP becomes

$$\begin{aligned} \Delta u(x) - \sigma u(x) &= \mathrm{f}(x) && \text{in } \Omega(\pi), \\ u(x) &= \mathrm{g}(x, \pi) && \text{on } \partial\Omega_\mathrm{D}(\pi), \\ \tfrac{\partial u}{\partial n}(x) - \mu u(x) &= \mathrm{h}(x) && \text{on } \partial\Omega_\mathrm{R}. \end{aligned} \tag{5}$$

Here $\partial/\partial n$ is the normal derivative; $\mu \in \mathbb{R}_{\geq 0}$ is a constant *Robin coefficient*; $\mathrm{h} : \partial\Omega_\mathrm{R} \to \mathbb{R}$ is the *Robin data*; for simplicity, we assume μ, h, and $\partial\Omega_\mathrm{R}$ do not depend on π. The solution to Equation (5) can be estimated via the *walk on stars (WoSt)* method [Miller et al. 2024], which is very similar to WoS but considers star-shaped domains rather than balls (see Miller et al. [2024, Algorithm 1] for details). We give a differential version of this estimator in Section 5.2.

4 PDE-CONSTRAINED SHAPE OPTIMIZATION

This section defines our central optimization problem (Equation (6)). We use classical results from shape optimization [Henrot and Pierre 2018] to express the derivatives in this problem as another BVP (Section 4.1), which we can then estimate via WoS (Section 5.1).

Shape functional. We consider a domain $\Omega(\pi)$ controlled by parameters π as in Section 3.1, and the solution $u(\cdot, \pi)$ to the primary BVP (3) on this domain. We want to determine values for π that locally minimize a *shape functional*

$$\mathcal{L}(\pi) := \int_{\Omega(\pi)} \mathrm{M}(x)\, \mathrm{L}(u(x, \pi))\, \mathrm{d}x. \tag{6}$$

Here $\mathrm{L} : \mathbb{R} \to \mathbb{R}$ is a differentiable *loss function*—for example, the squared loss $\mathrm{L}(u(x, \pi)) := 1/2 \|u(x, \pi) - u_\mathrm{ref}(x)\|^2$ relative to a reference solution u_ref. The function $\mathrm{M} : \mathbb{R}^3 \to \{0, 1\}$ is a *binary mask* that we use to localize the functional to a subdomain of $\Omega(\pi)$, such as a region of interest where a reference is available (see Section 7). Appendix C considers a more general functional that includes a boundary term, used in some of our examples (Section 7).

Algorithm 1 WalkOnSpheres(x, ε)

Input: A point x, and a termination parameter ε.
Output: A single-sample estimate $\widehat{u}(x)$ for Equation (3).

▷ Compute distance to and closest point on $\partial\Omega$
$\overline{\mathrm{R}}$, $\overline{x} \leftarrow$ DistanceToBoundary(x)
▷ Return boundary value g at $\overline{x}$ if $x \in \partial\Omega^{\varepsilon}$
if $\overline{\mathrm{R}} < \varepsilon$ **then return** $\mathrm{g}(\overline{x})$
▷ Uniformly sample a point w inside the unit ball
$w \leftarrow$ SampleUnitBall()
▷ Compute point for source contribution
$z \leftarrow x + \overline{\mathrm{R}} \cdot w$
▷ Compute source contribution
$\widehat{S} \leftarrow \mathrm{G}(\|z - x\|)/p^{\mathrm{B}}(x) \cdot \mathrm{f}(z)$
▷ Compute recursive contribution scale
$\alpha \leftarrow \mathrm{P}(\overline{\mathrm{R}})/p^{\partial\mathrm{B}}(x)$
▷ Attempt to terminate walk using Russian roulette
if $\alpha <$ SampleUniform(0, 1) **then return** $\widehat{S}$
▷ Uniformly sample a point v on the unit sphere
$v \leftarrow$ SampleUnitSphere()
▷ Compute next walk point
$x \leftarrow x + \overline{\mathrm{R}} \cdot v$
▷ Repeat procedure from next walk point
return WalkOnSpheres(x, ε) + $\widehat{S}$

Differentiating shape functionals. We minimize $\mathcal{L}(\pi)$ using *stochastic* gradient optimization, which requires stochastic estimates of the derivative of the shape functional $\mathcal{L}(\pi)$ with respect to the parameters π. From the Reynolds transport theorem, this derivative equals [Henrot and Pierre 2018, p. 239]:[2]

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\pi}(\pi) = \int_{\Omega(\pi)} \mathrm{M}(x)\, \dot{u}(x,\pi)\, \mathrm{L}'(u(x,\pi))\, \mathrm{d}x + \int_{\partial\Omega(\pi)} \mathrm{M}(x)\, \mathrm{V}_n(y,\pi)\, \mathrm{L}(u(y,\pi))\, \mathrm{d}y(y), \tag{7}$$

where L' is the derivative of the scalar loss L. Equation (7) is valid as long as the integrand of $\mathcal{L}$ does not have any π-dependent discontinuities—this assumption holds for all π-independent masking functions M and smooth loss functions L, as u is itself smooth.

Note that the boundary of Ω is always closed; in Section 8 we discuss how this formulation might be extended to open boundaries.

4.1 The differential boundary value problem

Evaluating Equation (7) requires computing the derivative $\dot{u}$ of the the solution u with respect to parameters π. In this section, we express $\dot{u}$ as the solution to a different BVP, which can in turn be estimated via a modified WoS algorithm (Section 5.1).

The primary BVP (3) implicitly defines u as a function of π. As explained by Henrot and Pierre [2018, Section 5.3], we can thus use *implicit differentiation* to obtain the derivatives $\dot{u}$ of this relationship. To this end, we first differentiate both sides of the screened Poisson equation with respect to π. Since the Laplacian Δ, screening coefficient σ, and source term f are independent of π, we get

$$\Delta\dot{u}(x) - \sigma\dot{u}(x) = 0 \quad \text{in } \Omega. \tag{8}$$

Implicit differentiation of the boundary condition depends on how we represent the boundary data g. We consider two cases.

Restricted boundary data. We first consider the case where g is the restriction of a scalar field $\mathrm{g}_{\mathbb{R}^3}(\cdot,\pi) : \mathbb{R}^3 \to \mathbb{R}$ to the domain boundary, *i.e.*, $\mathrm{g}(x) = \mathrm{g}_{\mathbb{R}^3}(x,\pi), \forall x \in \partial\Omega(\pi)$. Differentiating the boundary condition in Equation (3) with respect to π then yields

$$\dot{u}(x) = \dot{\mathrm{g}}_{\mathbb{R}^3}(x) + \left(\frac{\partial \mathrm{g}}{\partial n}(x) - \frac{\partial u}{\partial n}(x)\right)\mathrm{V}_n(x,\pi) \quad \text{on } \partial\Omega(\pi), \tag{9}$$

where $\partial\mathrm{g}/\partial n = n_x \cdot \nabla \mathrm{g}_{\mathbb{R}^3}$ (Appendix A).

Mapped boundary data. When we use an explicit representation for $\partial\Omega(\pi)$ (Section 3.1.2), we can alternatively define boundary values $\mathrm{g}_M(\cdot,\pi) : M \to \mathbb{R}$ on the reference surface M (possibly depending on π). For instance, if M is a mesh, then g_M could be defined via a texture map. These values are then pushed forward to $\partial\Omega(\pi)$ to obtain the current boundary data, *i.e.*, $\mathrm{g}(x) = \mathrm{g}_M(\Psi(x,\pi),\pi)$. In Appendix A we show that differentiating the boundary condition in Equation (3) yields

$$\dot{u}(x) = \dot{\mathrm{g}}_M(\Psi(x)) + \nabla \mathrm{g}_M(\Psi(x)) \cdot \dot{\Psi}(x) + \left(\frac{\partial \mathrm{g}}{\partial n}(x) - \frac{\partial u}{\partial n}(x)\right)\mathrm{V}_n(x,\pi) \quad \text{on } \partial\Omega(\pi), \tag{10}$$

where $\nabla \mathrm{g}_M$ is the spatial gradient of g_M on the reference surface, and $\partial\mathrm{g}/\partial n = \nabla \mathrm{g}_M \cdot (J_\Psi n_x)$, where J_Ψ is the Jacobian of Ψ.

Summary. Combining Equations (8)–(10), we can express the derivative $\dot{u}$ of the BVP solution u with respect to the parameters π as the solution to the following BVP:

Differential boundary value problem

$$\begin{aligned} \Delta\dot{u}(x) - \sigma\dot{u}(x) &= 0 && \text{in } \Omega(\pi), \\ \dot{u}(x) &= \mathrm{D}(x,\pi) - \mathrm{V}_n(x,\pi)\frac{\partial u}{\partial n}(x) && \text{on } \partial\Omega(\pi). \end{aligned} \tag{11}$$

In Equation (11), the function D can be inferred from the right-hand side of either Equation (9) or Equation (10). We call Equation (11) the *differential boundary value problem*. Comparing the differential BVP with the primary BVP (Equation (3)), we observe the following:

O1. Both BVPs solve the screened Poisson equation with Dirichlet boundary conditions in the same domain, differing only in their source term and boundary data.
O2. The two BVPs are *nested*, as the boundary data for $\dot{u}$ in the differential BVP depends on the solution u to the primary BVP.

These observations will facilitate the derivation of Monte Carlo estimators for the differential BVP in Section 5.1.

5 MONTE CARLO DERIVATIVE ESTIMATION

We now develop Monte Carlo algorithms for estimating the shape-functional derivative $\mathrm{d}\mathcal{L}/\mathrm{d}\pi$ in Equation (7). We do this in two parts: First, in Section 5.1, we introduce a differential walk on spheres

[2] Throughout, we distinguish between *total derivatives* $\mathrm{d}/\mathrm{d}\pi$ and *partial derivatives* $\partial/\partial\pi$.

algorithm for computing point estimates of $\dot{u}(\cdot, \pi)$. Then, in Section 5.3, we explain how to use these estimates to form Monte Carlo estimates of the domain and boundary integrals in Equation (7).

The differentiable rendering literature often refers to estimates of $\dot{u}$—and in particular their evaluation on a dense sampling of the domain $\Omega(\pi)$—as *forward-mode derivatives* [Nimier-David et al. 2020; Vicini et al. 2021; Zhang et al. 2023]. Our differential walk on spheres algorithm in Section 5.1 allows computing such derivatives, which we visualize in Figure 3, 6, and 8 to evaluate variance and bias. However, as we explain in Section 5.3, in optimization settings we can use our algorithm to directly compute so-called *reverse-mode (backward) derivatives* $\mathrm{d}\mathcal{L}/\mathrm{d}\pi$ with respect to all parameters π, without storing intermediate forward-mode derivatives. Doing so is analogous to the radiative backpropagation procedure by Nimier-David et al. [2020] and Vicini et al. [2021].

5.1 Differential walk on spheres

Observations O1 and O2 provide guidance for how to develop a Monte Carlo algorithm for estimating $\dot{u}$. Observation O1 shows that the WoS estimator we presented in Section 3.3, for the solution u to the primary BVP (3), can also estimate the solution $\dot{u}$ to the differential BVP (11), with a caveat: Observation O2 shows we must *nest* a WoS estimator for the primary BVP within one for the differential BVP—that is, allow the latter to use the outputs of the former.

The fact that (the normal derivative of) u only appears in the Dirichlet boundary data of the differential BVP Equation (11) makes this nesting straightforward: Once the WoS estimator for $\dot{u}$ reaches some point inside $\partial\Omega^\varepsilon(\pi)$, to compute the Dirichlet boundary data needed for termination, it invokes the WoS estimator for $\partial u/\partial n$, as we describe below. This nesting is equivalent to a process that starts a random walk for $\dot{u}$ and, at the point where the walk terminates, starts another random walk for $\partial u/\partial n$ (Figure 2).

Estimating normal derivatives. Complicating slightly this nesting procedure is the fact that the boundary condition of the differential BVP (11) uses not the solution u, but its normal derivative $\partial u/\partial n$ at a boundary point $x \in \partial\Omega(\pi)$. A modified WoS algorithm by Sawhney and Crane [2020, Section 3] can estimate spatial gradients; however, as Miller et al. [2023, Section 3.2] explain, it cannot do so at boundary points. Instead, Miller et al. approximate the normal derivative at x at an *offset point* $x - \ell \cdot n_x$ with $\ell > \varepsilon$, which they then estimate with the modified WoS algorithm [Sawhney and Crane 2020, Equation 13]. The result is the overall estimator

$$\widehat{\frac{\partial u}{\partial n}}(x) \overset{\text{[Miller et al. 2023]}}{:=} \widehat{\frac{\partial u}{\partial n}}(x - \ell \cdot n_x) = \frac{3}{\ell}(n_y \cdot n_x) \cdot \widehat{u}(y), \quad (12)$$

where the point y is sampled uniformly on the sphere $\partial\overline{\mathrm{B}}(x - \ell \cdot n_x)$.

We instead use backward differences to estimate the normal derivative through an estimate of the *solution* at the offset point as

$$\widehat{\frac{\partial u}{\partial n}}(x) := \frac{\mathrm{g}(x) - \widehat{u}(x - \ell \cdot n_x)}{\ell}. \quad (13)$$

In Equations (12) and (13), $\widehat{u}$ uses the WoS estimator of Equation (4). We found empirically (Figure 8) that our approach results in similar bias but less variance than the approach of Miller et al. [2023]. We suspect that the backward-differences estimator reduces variance by avoiding a cosine term over the first ball of the recursive walk.

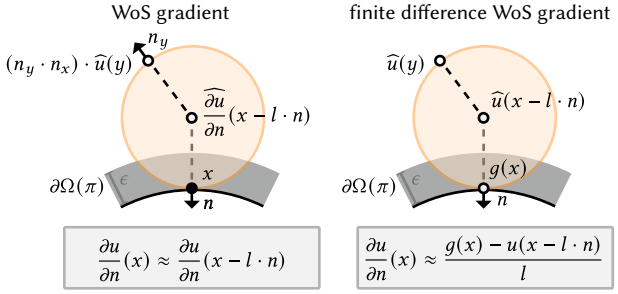


Figure 4. *Left:* The WoS estimator for the normal derivative by Sawhney and Crane [2020] in Equation (12) is undefined for points on the boundary, $x \in \partial\Omega(\pi)$, so Miller et al. [2023] evaluate it at an offset point $x - \ell \cdot n$. *Right:* We instead use the backward-difference approximation of the normal derivative in Equation (13) at the offset point.

A more comprehensive comparison of derivative estimators near the boundary, including off-centered gradient estimators [Yu et al. 2024, Equation 26], would be a useful future experiment.

Final estimator. Starting at the location $x_0 = x$ where we want to estimate $\dot{u}$, we can write our estimator recursively as follows:

Differential walk on spheres estimator

$$\widehat{\dot{u}}(x_k) := \begin{cases} \mathrm{D}(\overline{x_k}) - \frac{\mathrm{V}_n(\overline{x_k})}{\ell}(\mathrm{g}(\overline{x_k}) - \widehat{u}(\overline{x_k} - \ell n_{\overline{x_k}})), & x_k \in \partial\Omega^\epsilon, \\ \frac{\mathrm{P}(\overline{\mathrm{R}}(x_k))}{p^{\partial\mathrm{B}}(x_k)}\widehat{\dot{u}}(x_{k+1}), & \text{otherwise.} \end{cases} \quad (14)$$

In Equation (14), $\widehat{\dot{u}}$ is the WoS estimator of Equation (4). As in WoS, the next walk point x_{k+1} is sampled on the sphere $\partial\overline{\mathrm{B}}(x_k)$ with uniform probability $p^{\partial\mathrm{B}}(x_k)$. We term Equation (14) the *differential walk on spheres* estimator. As in WoS, we can also combine this estimator with Russian roulette for improved efficiency; we summarize the resulting procedure in Algorithm 2.

We conclude with two observations about the estimator in Equation (14): 1. Whereas at any point x the solution $u(x)$ is a scalar, the derivative $\dot{u}(x)$ is an N-dimensional vector, where N is the number of parameters π. Algorithm 2 estimates the derivatives with respect to all parameters π with a *single walk*. 2. Along with a single-sample estimate of $\dot{u}(x)$, Algorithm 2 can provide a single-sample estimate of $u(x)$ by returning the Dirichlet boundary condition at the terminal point $\mathrm{g}(\overline{x})$ in Line 11—effectively, using the same walk for both $\dot{u}(x)$ and $u(x)$. We do not make this modification explicit in Algorithm 2 to keep the basic algorithm simple, but we use it for estimation of reverse-mode derivatives in Section 5.3.

5.2 Differential walk on stars

We now return to the BVP (5) with mixed Dirichlet-Robin boundary conditions. We can derive a Monte Carlo estimator for the derivative $\dot{u}$ of its solution u with respect to parameters π, exactly analogously to how we did so for the primary BVP (3) in Sections 4.1 and 5.1. In

Algorithm 2 DiffWalkOnSpheres($x,\ \varepsilon,\ \ell$)

Input: A point x, a termination parameter ε, and an offset ℓ.
Output: A single-sample estimate $\widehat{u}(x)$ for Equation (11).

▷ Compute distance to and closest point on $\partial\Omega$
$\overline{\mathrm{R}},\ \overline{x} \leftarrow$ DistanceToBoundary(x)
▷ Compute and return boundary value at $\overline{x}$ if $x \in \partial\Omega^{\varepsilon}$
if $\overline{\mathrm{R}} < \varepsilon$ **then**
  ▷ Compute outward normal at $\overline{x}$
  $n \leftarrow$ BoundaryNormal($\overline{x}$)
  ▷ Compute offset point
  $o \leftarrow \overline{x} - \ell n$
  ▷ Estimate primary BVP solution at offset point
  $\widehat{u} \leftarrow$ WalkOnSpheres(o, ε)
  ▷ Return boundary value at $\overline{x}$
  return $\mathrm{D}(\overline{x}) - \mathrm{V}_n(\overline{x})(\mathrm{g}(\overline{x}) - \widehat{u})/\ell$
▷ Compute recursive contribution scale
$\alpha \leftarrow \mathrm{P}(\overline{\mathrm{R}})/p^{\partial\mathrm{B}}(x)$
▷ Attempt to terminate walk using Russian roulette
if $\alpha <$ SampleUniform(0, 1) **then return** 0
▷ Uniformly sample a point v on the unit sphere
$v \leftarrow$ SampleUnitSphere()
▷ Compute next walk point
$x \leftarrow x + \overline{\mathrm{R}} \cdot v$
▷ Repeat procedure from next walk point
return DiffWalkOnSpheres(x, ε, ℓ)

particular, $\dot{u}$ is the solution to the BVP [Henrot and Pierre 2018]

$$\begin{aligned} \Delta\dot{u}(x) - \sigma\dot{u}(x) &= 0 && \text{in } \Omega(\pi), \\ \dot{u}(x) &= \mathrm{D}(x,\pi) - \mathrm{V}_n(x,\pi)\frac{\partial u}{\partial n}(x) && \text{on } \partial\Omega_{\mathrm{D}}(\pi), \\ \frac{\partial\dot{u}}{\partial n}(x) - \mu\dot{u}(x) &= 0 && \text{on } \partial\Omega_{\mathrm{R}}. \end{aligned} \tag{15}$$

The observations O1 and O2 we made for the Dirichlet-only primary and differential BVPs (Equations (3) and (11)) also hold for their mixed Dirichlet-Robin counterparts (Equations (5) and (15)). Importantly, in the latter case nesting happens through *only* the Dirichlet boundary condition. Thus, we can estimate $\dot{u}$ using a differential WoSt algorithm that relates to the original WoSt exactly analogously to the relationship between differential and original WoS (Algorithms 1 and 2): Namely, the differential WoSt estimator performs a random walk to estimate $\dot{u}$, until it reaches a point in the ε-shell around $\partial\Omega_{\mathrm{D}}$; there, it invokes the WoSt estimator, which performs another random walk to estimate $\partial u/\partial n$. In Section 7.3, we use this differential WoSt algorithm in a thermal design setting.

We emphasize that the above approach applies only to the case where the Robin boundary condition in Equation (5) is independent of the parameters π we optimize. We leave parameterizing and differentiating the Robin boundary condition to future work, but discuss challenges with such an extension in Section 8.

5.3 Computing reverse-mode derivatives

We approximate the domain and boundary integrals in Equation (7) using Monte Carlo integration, by uniformly sampling points in the domain $\Omega(\pi)$ (for the first integral) or boundary $\partial\Omega(\pi)$ (for the second integral), and evaluating the corresponding integrands at those locations. Evaluating the integrand at different sample points $x \in \Omega(\pi)$ requires computing the loss function derivative L', which in turn requires computing values for both u and $\dot{u}$. The option to share walks between the primary and differential WoS estimators of these values creates opportunities for optimization. To elaborate, in the rest of this section, we specialize to the squared loss function $\mathrm{L}(u(x)) := 1/2\|u(x) - u_{\mathrm{ref}}(x)\|^2$. Then

$$\frac{\mathrm{d}\,\mathrm{L}}{\mathrm{d}\pi}(u(x)) = u(x)\dot{u}(x) - u_{\mathrm{ref}}(x)\dot{u}(x). \tag{16}$$

We discuss two baseline approaches to estimate Equation (16), and in particular the product $u\dot{u}$. Then we propose our own improved approach. For this discussion, we assume we have performed $2M$ walks of Algorithm 2, augmented as we described above so that each walk estimates both u and $\dot{u}$, with corresponding outputs $\{\widehat{u}_m, \widehat{\dot{u}}_m\}_{m=1}^{2M}$. We visualize all approaches in Figure 5 for the case $M = 1$.

Uncorrelated product estimator. The first baseline approach uses disjoint sets of walks—equivalently, separate calls to Algorithm 1 and Algorithm 2—to compute *independent* estimates $\widehat{u}$ and $\widehat{\dot{u}}$, then computes their product. We can express this approach as

$$\widehat{u} := \frac{1}{M}\sum_{m=1}^{M}\widehat{u}_m, \quad \widehat{\dot{u}} := \frac{1}{M}\sum_{m=M+1}^{2M}\widehat{\dot{u}}_m, \quad \langle u\dot{u}\rangle_{\mathrm{unc}} := \widehat{u}\cdot\widehat{\dot{u}}. \tag{17}$$

This approach is unbiased:[3] from independence, $\mathbb{E}[\langle u\dot{u}\rangle_{\mathrm{unc}}] = \mathbb{E}\big[\widehat{u}\big]\cdot\mathbb{E}\big[\widehat{\dot{u}}\big]$. However, it is sample-inefficient: it does not leverage the fact that each walk produces estimates for both u and $\dot{u}$, resulting in variance that scales as $O(1/M^2)$.

Correlated product estimator. The second baseline approach uses all walks to compute *correlated* estimates $\widehat{u}$ and $\widehat{\dot{u}}$, then again computes their product. We can express this approach as

$$\widehat{u} := \frac{1}{2M}\sum_{m=1}^{2M}\widehat{u}_m, \quad \widehat{\dot{u}} := \frac{1}{2M}\sum_{m=1}^{2M}\widehat{\dot{u}}_m, \quad \langle u\dot{u}\rangle_{\mathrm{cor}} := \widehat{u}\cdot\widehat{\dot{u}}. \tag{18}$$

Conversely to the uncorrelated approach, this approach is sample-efficient: it uses outputs for both u and $\dot{u}$ from all walks, and thus reduces variance at a faster rate $O(1/4M^2)$. However, it is biased: because of correlation, $\mathbb{E}[\langle u\dot{u}\rangle_{\mathrm{cor}}] \neq \mathbb{E}\big[\widehat{u}\big]\cdot\mathbb{E}\big[\widehat{\dot{u}}\big]$.

Our approach: U-statistic product estimator. To combine unbiasedness and sample efficiency, we introduce an approach that uses U-statistics—a methodology from statistics [Lee 1990] for combining estimators input to symmetric functions (*e.g.*, product) without introducing bias. Within computer graphics, Kettunen et al. [2021] used U-statistics to evaluate power-series transmittance estimators.

The U-statistic product estimator uses all pairwise combinations of estimates $\widehat{u}_m$ and $\widehat{\dot{u}}_{m'}$ such that $m \neq m'$

$$\langle u\dot{u}\rangle_{\text{U-stat}} := \frac{1}{2M(2M-1)}\sum_{m=1}^{2M}\sum_{\substack{m'=1\\ m'\neq m}}^{2M}\widehat{u}_m\widehat{\dot{u}}_{m'}. \tag{19}$$

Compared to the correlated estimator $\langle u\dot{u}\rangle_{\mathrm{corr}}$ in Equation (18), we observe that the U-statistic summation: 1. excludes only the

[3] More precisely, it does not introduce additional bias beyond that from the primary and differential WoS estimators (due to the ε-shell and normal-derivative approximations).

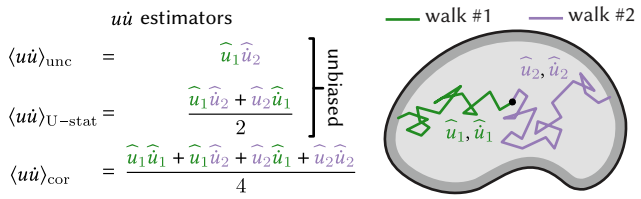


Figure 5. A Monte Carlo estimate of the product $\langle u\dot{u}\rangle$ requires estimating both the solution u and derivative $\dot{u}$. A single sampled walk simultaneously provides estimates of u and $\dot{u}$, but the product estimates $u\dot{u}$ are correlated and introduce bias. Rather than resort to uncorrelated estimates, which uses a single estimate from each walk, U-statistics shares complementary estimates across walks, which reduces variance without introducing bias.

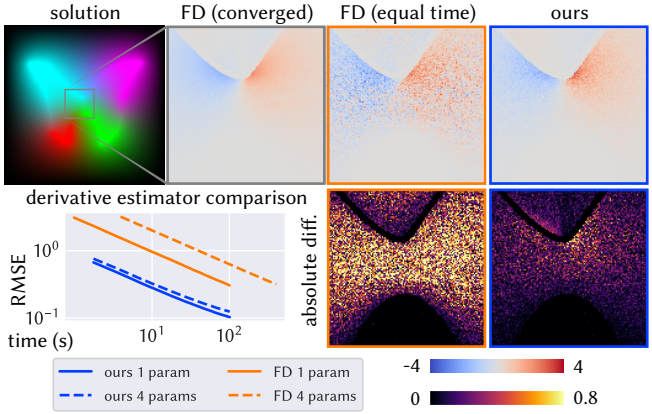


Figure 6. Finite-difference (FD) approximations of derivatives (*middle-right*) require only primary walks, but scale poorly with parameter count. Differential WoS (*far-right*) instead computes derivatives for all parameters π with a single differential walk, leading to less noisy results at equal time.

$2M$ correlated product terms $\widehat{u}_m\widehat{\dot{u}}_m$ to achieve unbiasedness, *i.e.*, $\mathbb{E}[\langle u\dot{u}\rangle_{\text{U-stat}}] = \mathbb{E}[\widehat{u}] \cdot \mathbb{E}[\widehat{\dot{u}}]$; 2. includes all other $2M(2M-1)$ product terms to achieve sample efficiency, reducing variance at a rate $O(1/4M^2)$. We can write Equation (19) equivalently as

$$\langle u\dot{u}\rangle_{\text{U-stat}} = \frac{1}{2M(2M-1)} \sum_{m=1}^{2M} \widehat{u}_m\left(S - \widehat{u}_m\right), \quad S := \sum_{m=1}^{2M} \widehat{u}_m. \quad (20)$$

Equation (20) provides a way to compute the U-statistic estimator in $O(2M)$ time, resulting in negligible computational overhead compared to the correlated and uncorrelated estimator. However, this estimator introduces a memory overhead, due to the need to store $\widehat{u}_m$ and $\widehat{\dot{u}}_m$ estimates as we sample walks. In our implementation, we strike a balance between sample efficiency and memory efficiency by applying U-statistics in batches of $B < 2M$ walks, computing Equation (20) after every $2M/B$ walks. In Section 6.3, we evaluate experimentally the performance of the U-statistic estimator compared to the uncorrelated and correlated ones, confirming it achieves sample efficiency while retaining unbiasedness.

Other loss functions. Our discussion assumed the squared loss function, but similar considerations apply to any other differentiable loss function L: namely, we can apply the uncorrelated, correlated, and U-statistic estimators to the product $\mathrm{d}\,\mathrm{L}/\mathrm{d}\pi = \dot{u}\,\mathrm{L}'(u)$, instead of $\dot{u}u$. In this case, none of the estimators will be unbiased, because in general $\mathbb{E}[\mathrm{L}'(u)] \neq \mathrm{L}'(\mathbb{E}[u])$. However, we expect that the relative performance of the three estimators in terms of bias and sample efficiency will remain the same as in the case of the squared loss.

Our discussion on estimation of products $\dot{u}\,\mathrm{L}'(u)$ for reverse-mode differentiation finds exact parallels in differentiable rendering, where such products appear when differentiating inverse rendering objectives. For example, Gkioulekas et al. [2016, Section 5] explain the need to use uncorrelated sets of paths for unbiased estimation of squared-loss derivatives; and Nicolet et al. [2023, Section 4] explain the bias when using other loss functions. These works also highlight the critical role of unbiasedness in ensuring convergence of stochastic gradient optimization, justifying our own focus on reducing bias while maintaining sample efficiency in reverse-mode derivatives. Given these parallels, our U-statistic estimator could potentially be useful also in differentiable rendering applications.

6 IMPLEMENTATION AND EVALUATION

We implement differential WoS on top of the open-source *Zombie* library [Sawhney and Miller 2023], without any changes to its WoS or WoSt routines for solving primary BVPs. For optimization, we use a combination of *vector Adam* [Ling et al. 2022] for shape parameters (*e.g.*, mesh vertices), and *Adam* [Kingma and Ba 2015] for all others parameters (*e.g.*, pose, rotation, color values). We use a constant learning rate of 10^{-3}, and exponential annealing of the number of walks per evaluation point (Section 6.2).

Before applying our solver to shape optimization problems in Section 7, in this section we evaluate several aspects of its performance: We first evaluate the accuracy of its derivative estimates against finite differencing (Section 6.1). We then study the impact of noisy derivatives on optimization (Section 6.2). We finally conduct an ablation study of hyperparameters and design choices (Section 6.3).

6.1 Comparison to finite differencing

To validate the accuracy of the derivative estimates output by our differential WoS estimator, we compare them in Figure 6 against estimates from finite differencing (specifically using forward differences). We make two observations: 1. When run till convergence, both techniques produce very similar outputs, thus providing evidence in support of the accuracy of our differential WoS estimator. 2. When run for equal time, differential WoS produces estimates of much lower RMSE than finite differencing. The performance improvement differential WoS provides becomes more pronounced as the number of parameters increases. The reason is that, to compute derivatives with respect to N parameters, finite differencing must perform $N+1$ independent primary walks; by contrast, differential WoS needs to perform only one differential walk irrespective of the value of N. The *only* overhead on differential WoS as N increases is storing a sparse vector of derivatives and computing, at the end of the walk, additional normal velocity terms $\mathrm{V}_n(x, \pi)$ for any parameter that influences the boundary at the terminal location.

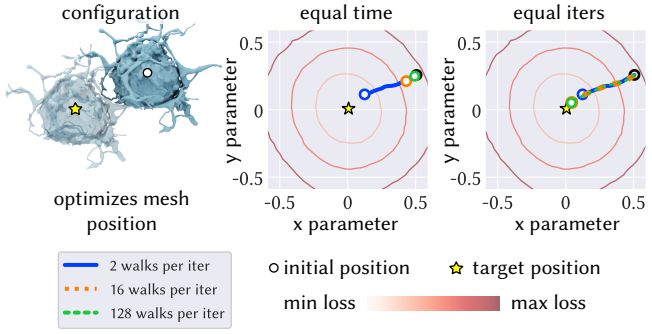


Figure 7. Monte Carlo estimation allow us to trade off between efficiency and noise during optimization by choosing the number of walks used for each derivative estimate. At equal time (*middle*), noisy derivatives make more progress towards the optimal parameters π, while at equal iterations (*right*), refined derivatives reach a more optimal solution.

Importantly, the number of WoS calls—or equivalently, the total number of walks—is constant with respect to N.

6.2 Optimization with noisy derivatives

In Figure 7, we show an experiment where we optimize the position of a mesh so as to match the solution to a screened Poisson equation on a nearby slice plane. We use this setting to evaluate how the variance of derivative estimates impacts optimization convergence and runtime. To this end, we optimize the mesh position by varying the number of differential walks per point (WPP) on the slice plane. Using a smaller WPP results in derivative estimates that are noisier yet also faster to compute; and conversely for larger WPP.

Our experiments show that, at *equal time*, using noisy derivatives (2 WPP) performs much better (*i.e.*, achieve faster progress towards the target mesh position) than using accurate derivatives (16 WPP or 128 WPP). On the other hand, at *equal number of iterations*, using more accurate derivatives converges faster than using noisy derivatives. This behavior highlights a critical advantage of stochastic gradient optimization, namely its ability to improve optimization runtime by using noisy (but unbiased) gradients to trade off between accuracy of descent directions versus iteration cost [Kushner and Yin 2003; Bottou and Bousquet 2008]. In practice, to ensure stable convergence, we follow Pfeiffer and Sato [2018] and progressively increase WPP using the annealing schedule

$$\mathrm{WPP}_t = \mathrm{WPP}_0 \cdot \exp(\log(\mathrm{WPP}_T/\mathrm{WPP}_0) \cdot t/N). \tag{21}$$

Here T is the number of total iterations, $t = 1, \ldots, T$ is the current iteration, and WPP_0 and WPP_T are the number of walks at the start and end of the optimization (resp.). This schedule provides fast-to-compute, but approximate descent directions in the early stages of the optimization where accuracy is not as important, and more accurate directions in the latter stages.

6.3 Ablations

In Figure 8, we ablate hyperparameters of differential WoS to compare the RMSE of estimated derivatives against reference derivatives computed using finite-differences. We use a 2D bunny mesh normalized to fit inside a unit sphere along with spatially varying Dirichlet boundary conditions, and compute the derivative $\dot{u}(x, \pi)$ with respect to a y-translation of the mesh.

ε-shell size. As Figure 8(a) shows, a larger ε value leads to lower variance but larger bias. We use $\varepsilon = 10^{-3}$ (corresponding to $1/1000\times$ the dimension of the scene's bounding cube) for all other experiments, as it provides a reasonable balance between bias and variance.

Normal derivative estimator. As Figure 8(b) shows, our backward-difference estimator (Equation (13)) achieves 20% lower RMSE than the estimator by Miller et al. [2023] (Equation (12), implemented in Zombie [Sawhney and Miller 2023] with antithetic sampling and control variates for reduced variance). We thus use the backward-difference estimator in all other experiments. We expect the performance difference between the two estimators to be smaller if the point $y \in \partial\overline{\mathrm{B}}$ in Equation (12) is sampled using cosine-weighted sampling [Pharr et al. 2023, Section 13.6.3].

Normal derivative offset. As Figure 8(c) shows, the offset ℓ in the backward-difference estimator (13) can result in increased bias if it is either too small or too large. In the extreme case when $\ell < \varepsilon$, WoS terminates immediately and returns the estimate $\hat{u}(x - \ell \cdot n_x) = \mathrm{g}(x)$, thus incorrectly always outputting zero normal derivative estimates. At $\ell = \varepsilon$, half of the first WoS steps will be inside the epsilon shell, which still leads to significant bias. Moving in the opposite direction, a large value $\ell = 100\times\varepsilon$ results in noticeable bias due to discretization error. Empirically, we found $\ell = 10 \times \varepsilon$ to perform well, and thus use this value for all other experiments.

U-statistic estimator. Figure 8(d) compares our U-statistic estimator (with a batch size $B = 8$) and the uncorrelated and correlated estimators from Section 5.3. Relative to the uncorrelated estimator, the correlated estimator increases bias but reduces variance, overall improving RMSE (which includes both bias and variance) by about $10 - 25\%$ depending on the WPP. Our U-statistic estimator achieves comparable RMSE to the correlated estimator, while remaining unbiased. We choose to use the U-statistic estimator in all other experiments, despite the fact that the correlated estimator slightly outperforms it in terms of RMSE. This choice is because prior work in differentiable rendering [Nicolet et al. 2023; Luan et al. 2021] has shown that low-variance yet biased derivative estimates can hinder convergence of stochastic gradient optimization.

7 GEOMETRIC EXAMPLES

We apply differential WoS on a range of PDE-constrained shape optimization tasks, including: diffusion-based pose estimation (Section 7.1) and surface reconstruction (Section 7.2), thermal design (Section 7.3), image-space curve inflation (Section 7.4), and inverse diffusion curves (Section 7.5). In most examples we optimize only boundary geometry, though we show joint optimization of boundary geometry and data in Section 7.5. We report detailed statistics for each optimization task in Table 1. The project website includes: 1. a video file visualizing the optimization process for all of our examples; and 2. code for our implementation.

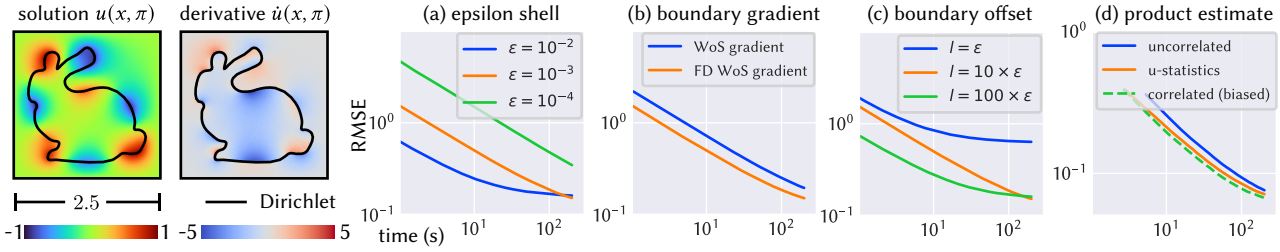


Figure 8. We ablate various hyperparameters of differential WoS on a 2D scene with spatially varying Dirichlet boundary conditions, to study the convergence of derivative estimates. Derivatives are computed with respect to a translation of the mesh in the y-direction.

7.1 Pose estimation

Background. Pose estimation seeks to recover the position, orientation, and scale of a known shape. A challenging subset of this problem uses measurements of light partially obscured by scattering on a diffusive surface [Isogawa et al. 2020] or through a diffusive medium [Raviv et al. 2014]. We consider the latter scenario in Figure 9, where we approximate radiative transport in diffusive media with a screened Poisson equation [Wang and Wu 2007, Ch. 5].

Setup. In Figure 9, we model a fixed-shape emissive object as a Dirichlet boundary, and optimize seven pose parameters: four for rotation and scale (Euler-Rodrigues parameterization), and three for translation. The Dirichlet boundary influences the image plane measurements via the primary BVP (3) with f = 0 and g = 1.

Output-sensitive optimization. Our differential WoS algorithm can efficiently optimize shape functionals (6) that are localized, through the masking function M, in small subsets of the domain $\Omega(\pi)$: As a pointwise estimator, differential WoS can estimate gradients only at locations where M = 1, unlike conventional grid-based solvers that are constrained to output derivatives over the entire domain—thus wasting significant compute. This output sensitivity of differential WoS makes it well-suited to application settings where we can only measure physical quantities (*e.g.*, light, heat, electric potentials) locally. To demonstrate output sensitivity, in Figure 9 we optimize pose by minimizing a shape functional that uses the squared loss L and a mask M to compare with a reference solution only on an image plane (Figure 9, top right corner). Differential WoS allows us to focus all computation only on this image plane.

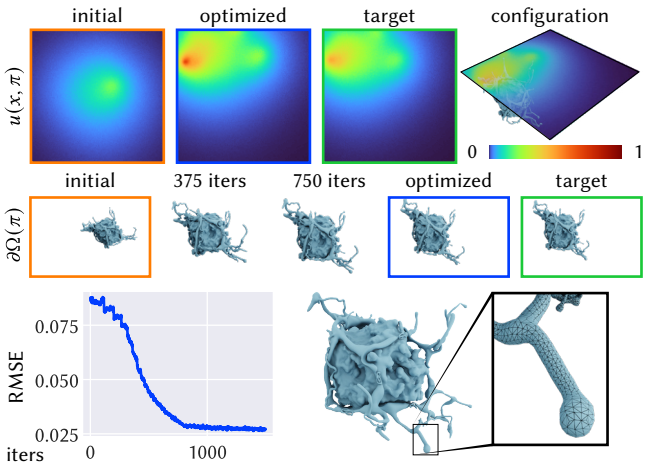


Figure 9. We recover the rotation, position, and scale of a complex boundary mesh (116k faces) to match a solution profile on an image plane (*top right*).

7.2 Shape from diffusion

Background. Differentiable Monte Carlo rendering algorithms [Zhang et al. 2023], alongside effective preconditioners [Nicolet et al. 2021], have recently shown great promise for shape reconstruction tasks from radiometric images, providing benefits such as geometric robustness and scalability, output sensitivity, and trivial parallelism. Differential WoS can bring the same benefits to shape reconstruction tasks from measurements of diffusive phenomena that can be modeled via elliptic PDEs. Such measurements can arise in medical imaging applications due to deep tissue scattering [Schweiger et al. 1995; Arridge 1999], or from alternative imaging modalities such as thermal imaging [Harbrecht and Tausch 2013]. In Figures 1 and 10, we show proof-of-concept experiments for such shape-from-diffusion tasks.

Setup. As in the pose estimation experiment, we model an emissive object of unknown shape as a Dirichlet boundary with constant Dirichlet data. We represent the object as triangular mesh, initialized to a spherical shape. We optimize the mesh vertices to match a reference solution specified on the six faces of a bounding box, as in Figure 1 (top right). The emissive object influences the solution on the bounding box via the primary BVP (3) with f = 0 and g = 1.

Gradient preconditioning. During optimization, we use the Laplacian gradient preconditioner from Nicolet et al. [2021], and remesh every 100 iterations [Botsch and Kobbelt 2004]. In addition to Vector Adam, we also follow the suggestion of Nicolet et al. [2021] to adapt step size using the maximum momentum across all parameters.

High-frequency details. Surface details are faithfully reconstructed to the extent to which they influence diffusive measurements. Due to diffusion, fine geometry details have little influence on the solution at points away from the emissive boundary. We showcase this phenomenon in Figure 10 (*bottom row*): reconstructing a low-frequency approximation to the ground truth high-frequency sphere geometry perfectly reproduces the reference solution.

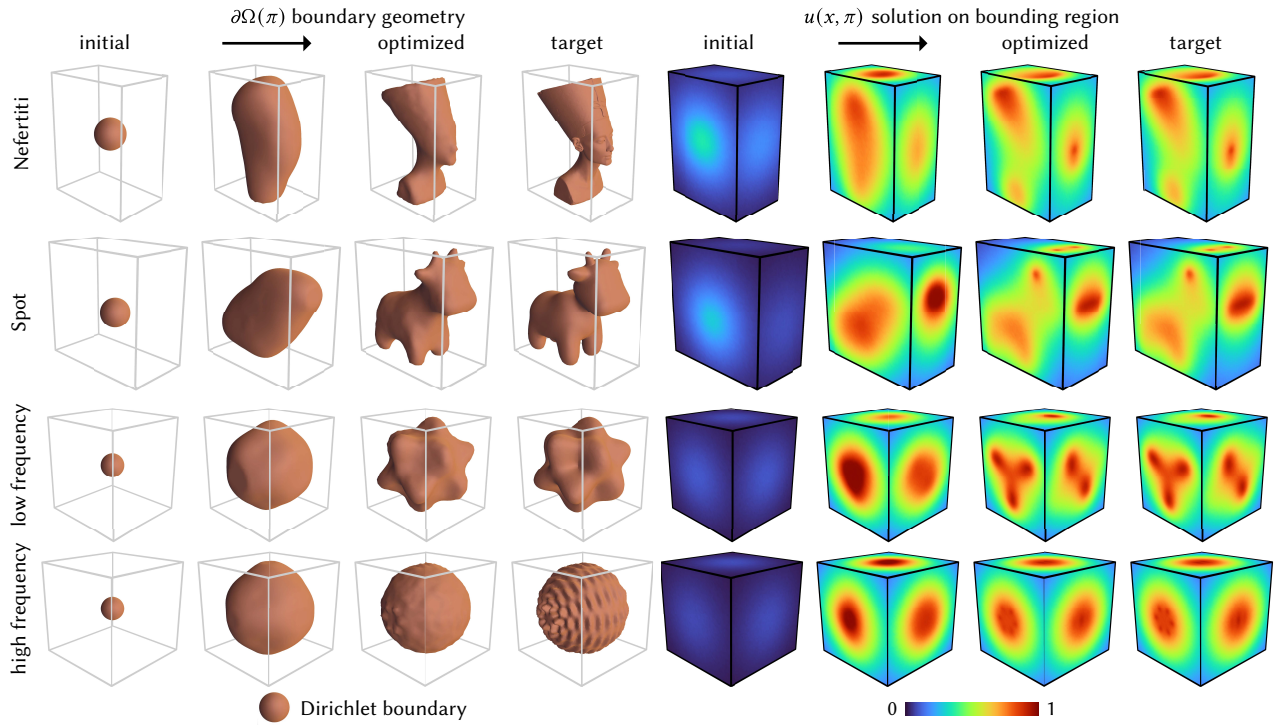


Figure 10. Inspired by shape-from-diffusion applications, we use measurements on a bounding box (*far right, column*) to evolve 3D models starting from a sphere (*left*). As with any reconstruction through a diffusive medium, high-frequency surface details have negligible influence on the solution, and are therefore not captured in the reconstructions (*bottom row*).

7.3 Optimization-driven thermal design

Background. Elliptic PDEs are at the heart of classic optimization-driven design problems ranging from finding airfoil shapes that minimize drag at cruising speeds, to constructing electromagnets that induce fields with specific characteristics [Pironneau 1984]. Recently, thermal systems have become a major focus of optimization-driven design [Zhan et al. 2008; Dbouk 2017], as the shape of components can critically impact, for instance, the performance of integrated circuits or the effectiveness of heat exchangers.

Setup. In Figure 11, we optimize toaster heating elements to achieve a target temperature profile on a slice of bread. Following Sawhney et al. [2023, Section 6.5], we model conductive heat transfer from the toaster wires (corresponding to a parameterized Dirichlet boundary $\partial\Omega_{\mathrm{D}}(\pi)$ with 500 line segments) to the slice of bread (corresponding to a fixed Neumann boundary $\partial\Omega_{\mathrm{R}}$ with 250k faces), by solving the BVP (5) with $\mathrm{f} = 0$, $\mathrm{g} = 1$, $\mu = 0$, and $\mathrm{h} = 0$. As this is a mixed Neumann-Dirichlet BVP, we use the differential WoSt algorithm of Section 5.2 for optimization.

Similar to Sawhney et al. [2023], we visualize the solution using a phenomenological model that maps temperature $u(x,\pi)$ to color values. We further constrain the heating elements by regularizing their polyline geometry with a bending and stretching energy [Bergou et al. 2008]. Though we cannot achieve arbitrarily sharp temperature profiles due to the physical constraints of diffusive heat transfer, we find the closest physically realizable profiles.

Output-sensitive optimization. The heating elements are optimized with respect to a temperature loss defined on the Neumann boundary $\partial\Omega_{\mathrm{R}}$. To determine the reference temperature on $\partial\Omega_{\mathrm{R}}$, we project a target temperature profile u_{ref} from an image plane $\mathcal{S}$ onto $\partial\Omega_{\mathrm{R}}$ via ray tracing—we denote this projection operation as $\mathbf{proj}(x)$. Doing so, we can avoid directly integrating our loss over all 250k faces on $\partial\Omega_{\mathrm{R}}$, and instead integrate it over just the image plane,

$$\mathcal{L}(\pi) := \int_{\mathcal{S}} \|u(\mathbf{proj}(x),\pi) - u_{\mathrm{ref}}(x)\|^2 \,\mathrm{d}x \tag{22}$$

This "deferred shading" approach evaluates derivatives at the minimum resolution necessary to capture the details in the target temperature profile (*i.e.*, 128^2 resolution grid), and only from visible locations on $\partial\Omega_{\mathrm{R}}$ (*i.e.*, the front of the slice of bread). Compared to a naive evaluation at 124k mesh faces of $\partial\Omega_{\mathrm{R}}$, deferred shading reduces the number of evaluation points by an order of magnitude.

Local modifications. When updating a temperature profile resembling the word "ham" to the word "jam", we initialize with the heating elements optimized for "ham". Output-sensitive optimization lets us optimize with respect to the loss only over the letter "j", which reduces average iteration time by over 80%.

7.4 Image-space curve inflation

Background. Many prior works have observed the central relationship between silhouettes and 3D shape [Ikeuchi and Horn 1981; Cipolla and Giblin 2000; DeCarlo et al. 2003], which allows depth

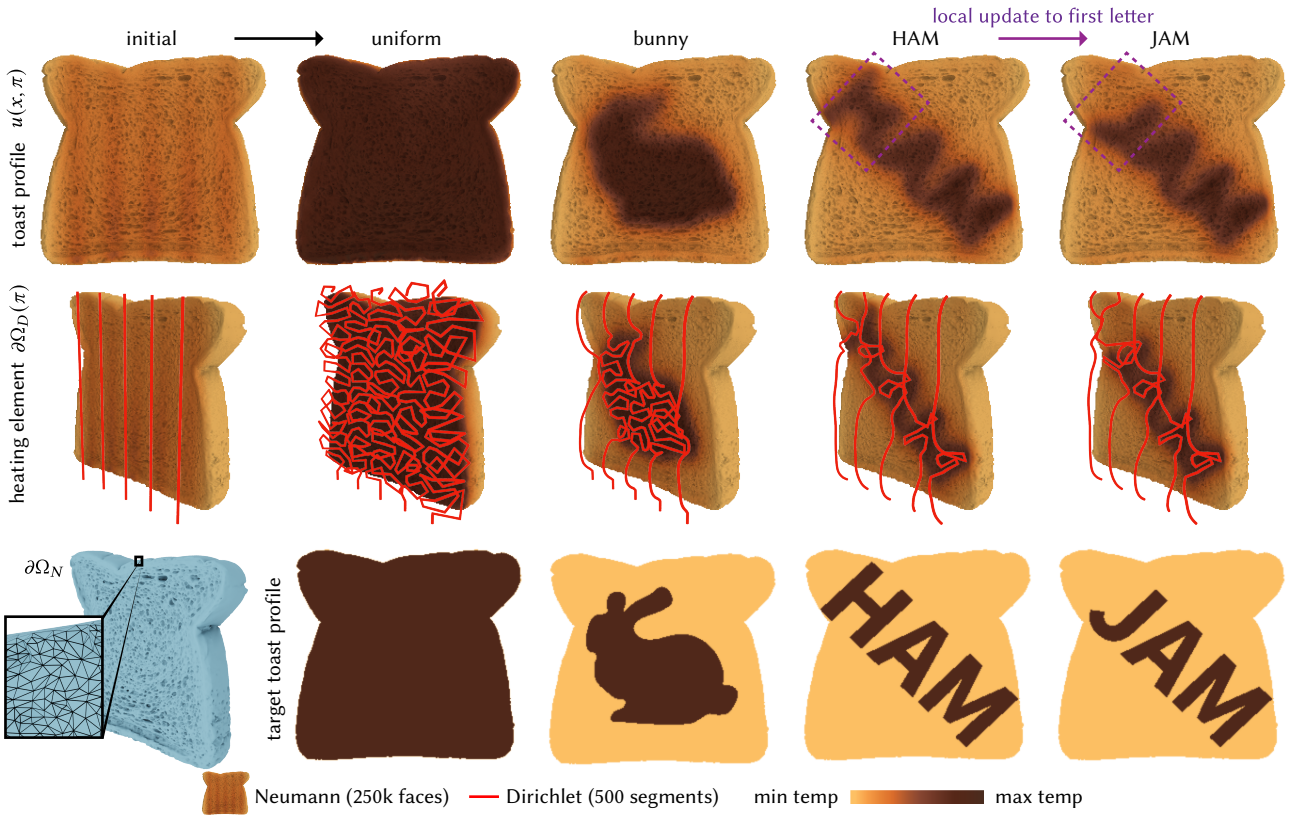


Figure 11. We optimize the heating elements of a toaster (Dirichlet boundary, *middle row*) to achieve a target temperature profile (*bottom row*) on a slice of bread (Neumann boundary, *top row*). For partial updates to a target profile (*e.g.*"ham" to "jam", *bottom right*), we leverage locality-sensitive optimization by computing derivatives only over the modified profile.

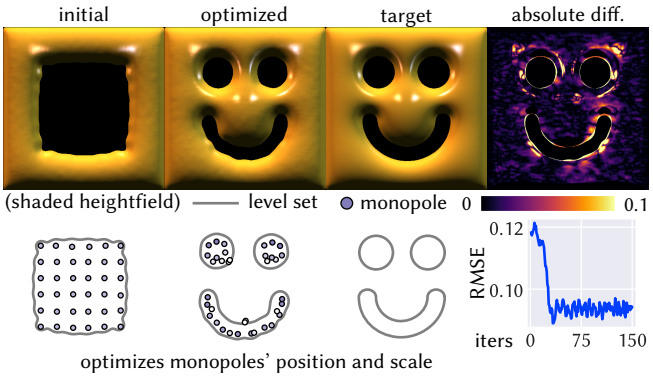


Figure 12. Baran and Lehtinen [2009] define smooth heightfields over a domain as the solution to a Poisson equation with zero Dirichlet boundary conditions. We optimize an implicit boundary, specifically the position and scale of harmonic radial basis functions, to achieve a target heightfield visualized as a shaded surface (*top row*).

to be modeled in image space as the solution to a Poisson equation with boundary conditions imposed along silhouettes [Ikeuchi and Horn 1981; Gorelick et al. 2006]. Inspired by this relationship, Baran and Lehtinen [2009] propose an artistic *curve inflation* model where height is modeled as a solution to the BVP Equation (3) with $\sigma = 0$, $f = 4$, and $g = 0$. Intuitively, the height u is zero on the boundary, and is smoothly "inflated" into the interior of $\Omega(\pi)$. Thus controlling the boundary of the BVP allows controlling the resulting height field. We consider the inverse problem in Figure 12, where we recover the boundary from a heightfield produced with curve inflation.

Setup. In Figure 12, we optimize the boundary $\partial\Omega(\pi)$ to match a reference heightfield $u_{\text{ref}}(x)$—which we visualize as a shaded surface. We represent the boundary $\partial\Omega(\pi)$ *implicitly* as the zero-level set of a function $h(x,\pi)$ parameterized as the sum of N harmonic monopoles—each with a scale $a_n \in \mathbb{R}$ and position $p_n \in \mathbb{R}^2$

$$h(x,\pi) := c + \sum_{n=1}^{N} \frac{a_n}{\|x - p_n\|}, \quad \partial\Omega(\pi) := \left\{x \in \mathbb{R}^2 : h(x,\pi) = 0\right\}. \quad (23)$$

By using an implicit representation, we can easily handle large topological changes to the boundary, as we show in Figure 12. Applying differential WoS in this setting requires simply implementing conservative closest point queries for this implicit representation. As h is a *harmonic function*, we can achieve this using *Harnack bounds* [Gillespie et al. 2024]: these bounds provide a conservative radius

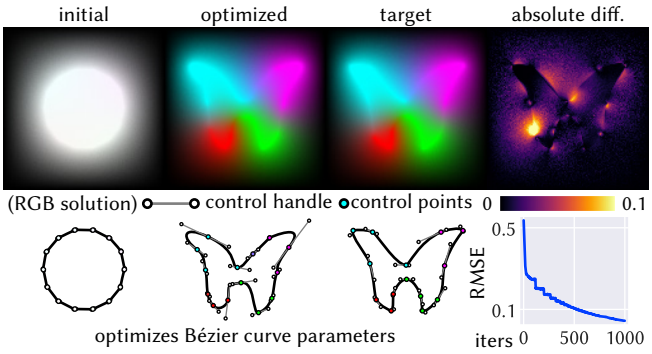


Figure 13. Inspired by Zhao et al. [2017], we optimize Bézier curves and RGB boundary conditions to generate a diffusion solution that matches a target image. Unique to differential WoS, we can directly optimize the control points and handles of the Bézier curves.

for an empty sphere at any point, as well as a gradient-sensitive condition $|h(x)| < |\nabla h(x)|\varepsilon$ for terminating walks at an ε-shell.

Implicit boundary integrals. Direct sampling of the boundary integral in the shape functional derivative (7) is not feasible in this example, because the boundary is defined implicitly. We instead approximate the boundary integral as a volume integral,

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\pi}(\pi) \approx \int_{\Omega(\pi)\setminus\partial\Omega_\varepsilon(\pi)} \dot{u}(x,\pi)\,\mathrm{L}'(u(x,\pi))\,\mathrm{d}x + \frac{1}{2\varepsilon}\int_{\partial\Omega_\varepsilon(\pi)} \mathrm{V}_n(x,\pi)\,\mathrm{L}(u(x,\pi))\,\mathrm{d}x, \tag{24}$$

where $\partial\Omega_\varepsilon(\pi) \coloneqq \{x \in \Omega(\pi) : \min_{y\in\partial\Omega(\pi)} \|x-y\| < \varepsilon\}$. In the limit of $\epsilon \to 0$ we recover the original boundary integral.

7.5 Inverse diffusion curves

Background. Orzan et al. [2013] introduced *diffusion curves* as a vector graphic primitive that models images as the solution $u(x) : \Omega \to [0,1]^3$ to a Laplace equation—that is, the BVP Equation (3) with $\sigma = 0$ and $\mathrm{f} = 0$—with RGB-valued Dirichlet boundary conditions $\mathrm{g}(\cdot,\pi)$ along a boundary $\partial\Omega(\pi)$ represented as Bézier curves. Rather than manually create diffusion curves, Zhao et al. [2017] optimize $\partial\Omega(\pi)$ and $\mathrm{g}(\cdot,\pi)$ to match a target image. They use FEM to solve Equation (3), which constrains them to use a polyline boundary representation. As we show in Figure 13, with differential WoS we can instead optimize a boundary encoded using Bézier curves.

Setup. We jointly optimize over both the Bézier curves $\partial\Omega(\pi)$ and the RGB-valued Dirichlet boundary conditions $\mathrm{g}(\cdot,\pi)$ to match a target image. We enforce G^1 continuity connections between the curves, by parameterizing the boundary so that adjacent Béziers have co-linear control handles. Our optimization also accounts for continuous Dirichlet conditions arising from linear interpolation of RGB values between control points.

Monte Carlo length regularization. We follow Zhao et al. [2017, Section 5.3] and use a length regularizer $R(\pi) \coloneqq \alpha \int_{\partial\Omega(\pi)} \mathrm{d}l(x)$ to penalize geometrically complex curves. During optimization, we use Monte Carlo estimation of the derivative of this regularizer:

$$\dot{R}(\partial\Omega(\pi)) = \alpha \int_{\partial\Omega(\pi)} \kappa(x)\,\mathrm{V}_n(x)\,\mathrm{d}l(x) \tag{25}$$

where κ is the curvature of the Bézier curve (available in closed form). To ensure the regularization is scaled appropriately relative to estimator variance, we decay its strength using a schedule $\alpha_t \coloneqq \alpha_0(\mathrm{WPP}_0/\mathrm{WPP}_t)^{-1/2}$, where α_t and WPP_t are the regularization strength and WPP (resp.) at the t^{th} gradient iteration. We choose the decay rate of the schedule to match the expected Monte Carlo convergence rate of $O(n^{-1/2})$.

8 CONCLUSION AND FUTURE WORK

We introduced a differential walk on spheres method to estimate derivatives with respect to perturbations of the domain boundary. In developing this method, we crucially take the approach of differentiating the functional relationship in Equation (3) rather than the estimator in Equation (4). This approach allows us to easily adapt the the classic WoS method, used for solving primary PDEs [Muller 1956; Sawhney and Crane 2020], into a method for solving a set of nested BVPs for the parameter derivative.

Even though these nested BVPs are well known in the shape optimization literature [Henrot and Pierre 2018], using WoS to estimate them brings to PDE-constrained shape optimization the unique benefits of Monte Carlo methods—output-sensitive evaluation, geometric robustness and flexibility, and the scalability of stochastic gradient-based optimization. We illustrated these benefits through varied geometric examples, yet our method is only a first step towards applying Monte Carlo PDE solvers to real-world inverse problems. To help further this goal, we discuss generalizations and improvements for enhancing the capabilities of our method.

More sophisticated physical models. We focused on the screened Poisson equation due to its conceptual simplicity and widespread use in graphics and other sciences. This PDE already places our technique within scope for inverse problems used in practice. For instance, medical imaging with electrical impedance tomography [Cheney et al. 1999] involves optimizing a variable diffusion coefficient in a screened Poisson equation with Robin boundary conditions [Uhlmann 2009]—recent work generalizes WoS to PDEs of this form [Sawhney et al. 2022; Miller et al. 2024]. Thermal imaging techniques like shape from heat [Chen et al. 2015] require simulating both the propagation of infrared light and the transfer of heat through solids via conduction. Already Bati et al. [2023] have developed such a Monte Carlo solver for coupled heat transfer; a differential variant could be derived by considering both differentiable rendering and differential WoS. However, more general inverse problems require extending our technique to more general PDEs. For example, the design of heat exchangers [Feppon 2019] involves simulating the coupled physics of convection and conduction, often via simplified models for the flow of fluids and heat diffusion (resp.)—simulation of fluids in a Monte Carlo framework with WoS (and related solvers) has recently garnered attention in graphics [Rioux-Lavoie et al. 2022; Jain et al. 2024; Sugimoto et al. 2024].

Parameterized Robin boundaries. We discussed in Section 5.2 how WoSt [Sawhney et al. 2023; Miller et al. 2024] can be used, in a limited

Table 1. We report the average iteration time of differential WoS across a variety of applications. For certain applications, we increase the number of differential walks per point on an exponential schedule (indicated by "initial → final" in the table). The number of differential walks lists the walks per iteration, whereas the number of primary walks lists the number of recursive walks to estimate the differential boundary condition, for each differential walk. All applications were run on a 12 core i9-10920X Intel CPU, except for curve inflation which was prototyped on a NVIDIA 3090 GPU.

application	geometry $\partial\Omega(\pi)$	parameters π	# eval pts	# diff. walks	# primary walks	# iters	avg. iter time (s)
pose estimation	mesh (58k vertices)	pose matrix	128^2	$2 \rightarrow 64$	2	1.5k	8s
surface reconstruction	mesh (2.5k vertices)	vertex pos.	6×64^2	$16 \rightarrow 256$	16	1k	82s
thermal design	polylines (500 vertices)	vertex pos.	128^2	$4 \rightarrow 16$	8	250	12s
curve inflation	implicit (32 monopoles)	pos. & scale	128^2	4	4	150	2s
diffusion curves	Béziers (14 curves)	pos. & tangents	128^2	$2 \rightarrow 64$	2	1k	3s

capacity, to optimize the Dirichlet boundary in a mixed Dirichlet-Robin BVP. Optimizing parameterized Robin boundaries $\partial\Omega_R(\pi)$ is likewise possible by evaluating a corresponding nested Dirichlet-Robin BVP [Henrot and Pierre 2018, Page 230, Eq 5.79]. In this case, nesting occurs via a differential Robin boundary condition evaluated at each step of the differential WoSt estimator. However unlike Equation (15), this boundary condition involves evaluating second-order spatial gradients that can lead to high variance—lowering noise of such gradients is necessary to make this approach practical.

Open boundaries. In Section 4 we restricted our discussion to closed boundaries, and this restriction is a current limitation of our method. The reason for this restriction is that, when working with an open boundary, the differential BVP will include a boundary condition with a Dirac delta term on the boundary perimeter. WoS cannot importance sample such localized boundary conditions, and thus this limitation also constrains differential WoS. This limitation could be overcome by using alternative bidirectional solvers [Qi et al. 2022; Miller et al. 2024] that simulate *reverse* random walks starting from the boundary. Support for open boundaries should be possible by using reverse differential walks with our method.

Variance reduction. More generally, any improvement in estimation quality to the underlying WoS estimator should translate directly to differential WoS—this includes recently developed variance reduction strategies for WoS [Qi et al. 2022; Miller et al. 2023; Bakbouk and Peers 2023; Li et al. 2023] we did not consider in this paper for simplicity. Specific to geometric optimization, it is also worthwhile investigating how to reuse derivative estimates with similar values across successive iterations in an optimization to improve efficiency; such strategies have found success in differentiable Monte Carlo rendering, in the form of recursive control variates [Nicolet et al. 2023] and reservoir-based temporal importance resampling [Chang et al. 2023]. Finally, we should improve upon the basic finite difference scheme we use in Section 5.1 to lower both the variance and bias in our normal derivative estimates—the alternative normal derivative estimator developed by Yu et al. [2024] applies directly to our method as well.

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 and the authors thank Gilles Daviet for helpful discussion.

REFERENCES

Grégoire Allaire, Charles Dapogny, and Pascal Frey. 2013. A mesh evolution algorithm based on the level set method for geometry and topology optimization. *Structural and Multidisciplinary Optimization* 48 (2013), 711–715.

Grégoire Allaire, Charles Dapogny, and Pascal Frey. 2014. Shape optimization with a level set based mesh evolution method. *Computer Methods in Applied Mechanics and Engineering* 282 (2014), 22–53.

Grégoire Allaire and Olivier Pantz. 2006. Structural Optimization with FreeFem++. *Structural and Multidisciplinary Optimization* 32 (2006), 173–181. https://hal.science/hal-01089081

Hend Ben Ameur, Martin Burger, and Benjamin Hackl. 2004. Level set methods for geometric inverse problems in linear elasticity. *Inverse Problems* 20, 3 (2004), 673.

Simon R Arridge. 1999. Optical tomography in medical imaging. *Inverse problems* (1999).

Ghada Bakbouk and Pieter Peers. 2023. Mean Value Caching for Walk on Spheres. In *Eurographics Symposium on Rendering*. The Eurographics Association. https://doi.org/10.2312/sr.20231120

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.* 39, 6, Article 245 (nov 2020), 18 pages. https://doi.org/10.1145/3414685.3417833

Ilya Baran and Jaakko Lehtinen. 2009. *Notes on inflating curves.* Technical Report. MIT.

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, Article 79 (jul 2023), 20 pages. https://doi.org/10.1145/3592121

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete Elastic Rods. *ACM Transactions on Graphics (SIGGRAPH)* 27, 3 (aug 2008), 63:1–63:12.

Mario Botsch and Leif Kobbelt. 2004. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Nice, France) *(SGP '04)*. Association for Computing Machinery, New York, NY, USA, 185–192. https://doi.org/10.1145/1057432.1057457

Leon Bottou and Olivier Bousquet. 2008. The Tradeoffs of Large Scale Learning. *NIPS* (2008).

G. Cai, K. Yan, Z. Dong, I. Gkioulekas, and S. Zhao. 2022. Physics-Based Inverse Rendering using Combined Implicit and Explicit Geometries. *Computer Graphics Forum* 41, 4 (2022).

Jean Céa, Alain Gioan, and Jean Michel. 1973. Quelques résultats sur l'identification de domaines. *Calcolo* 10 (1973), 207–232.

Jean Cea, Alain Gioan, and Jean Michel. 1974. Adaptation de la méthode du gradient à un problème d'identification de domaine. In *Computing Methods in Applied Sciences and Engineering Part 2: International Symposium, Versailles, December 17–21, 1973*. Springer, 391–402.

Wesley Chang, Venkataram Sivaram, Derek Nowrouzezahrai, Toshiya Hachisuka, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Parameter-space ReSTIR for Differentiable and Inverse Rendering. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, 10. https://doi.org/10.1145/3588432.3591512

Chia-Yen Chen, Chia-Hung Yeh, Bao Rong Chang, and Jun-Ming Pan. 2015. 3D Reconstruction from IR Thermal Images and Reprojective Evaluations. *Mathematical problems in engineering* 2015 (2015), 1–8.

Margaret Cheney, David Isaacson, and Jonathan C Newell. 1999. Electrical impedance tomography. *SIAM review* 41, 1 (1999), 85–101.

Roberto Cipolla and Peter Giblin. 2000. *Visual Motion of Curves and Surfaces*. Cambridge University Press.

Talib Dbouk. 2017. A review about the engineering design of optimal heat transfer systems using topology optimization. *Applied Thermal Engineering* 112 (2017), 841–854. https://doi.org/10.1016/j.applthermaleng.2016.10.134

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 (MIG '23)*. Association for Computing Machinery, New York, NY, USA, Article 9, 7 pages. https://doi.org/10.1145/3623264.3624457

Douglas DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. 2003. Suggestive contours for conveying shape. *ACM SIGGRAPH 2003 Papers* (2003). https://api.semanticscholar.org/CorpusID:1485904

Tao Du, Kui Wu, Andrew Spielberg, Wojciech Matusik, Bo Zhu, and Eftychios Sifakis. 2020. Functional optimization of fluidic devices with differentiable stokes flow. *ACM Trans. Graph.* 39, 6, Article 197 (nov 2020), 15 pages. https://doi.org/10.1145/3414685.3417795

Florian Feppon. 2019. *Shape and topology optimization of multiphysics systems*. Ph.D. Dissertation. Université Paris-Saclay (ComUE).

Mark Gillespie, Denise Yang, Mario Botsch, and Keenan Crane. 2024. Ray Tracing Harmonic Functions. *ACM Trans. Graph.* 43, 4 (2024).

Ioannis Gkioulekas, Anat Levin, and Todd Zickler. 2016. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer, 685–701.

Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. 2013. Inverse Volume Rendering with Material Dictionaries. *ACM Trans. Graph.* 32, 6 (2013), 162:1–162:13.

Lena Gorelick, Meirav Galun, Eitan Sharon, Ronen Basri, and Achi Brandt. 2006. Shape Representation and Classification Using the Poisson Equation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 12 (2006), 1991–2005. https://doi.org/10.1109/TPAMI.2006.253

Pavel Grinfeld. 2013. *Introduction to tensor analysis and the calculus of moving surfaces*. Springer.

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099* (2020).

Jacques Hadamard. 1908. *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*. Vol. 33. Imprimerie nationale.

Helmut Harbrecht and Johannes Tausch. 2013. On the Numerical Solution of a Shape Optimization Problem for the Heat Equation. *SIAM Journal on Scientific Computing* 35, 1 (2013), A104–A121. https://doi.org/10.1137/110855703 arXiv:https://doi.org/10.1137/110855703

Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper_files/paper/2022/file/8fcb27984bf16ca03cad643244ec470d-Paper-Conference.pdf

Antoine Henrot and Michel Pierre. 2018. *Shape variation and optimization*.

Raymond M. Hicks and Preston A. Henne. 1977. Wing design by numerical optimization. In *American Institute of Aeronautics and Astronautics, Aircraft Systems and Technology Meeting, Seattle, Wash.; United States; 22-24 Aug. 1977*.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2020. DiffTaichi: Differentiable Programming for Physical Simulation. arXiv:cs.LG/1910.00935

Katsushi Ikeuchi and Berthold KP Horn. 1981. Numerical shape from shading and occluding boundaries. *Artificial intelligence* 17, 1-3 (1981), 141–184.

Mariko Isogawa, Ye Yuan, Matthew O'Toole, and Kris M. Kitani. 2020. Optical Non-Line-of-Sight Physics-Based 3D Human Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pranav Jain, Ziyan Qu, Peter Yichen Chen, and Oded Stein. 2024. Neural Monte Carlo Fluid Simulation. *ACM Trans. Graph.* 43 (July 2024). https://doi.org/10.1145/3641519.3657438

Markus Kettunen, Eugene D'Eon, Jacopo Pantaleoni, and Jan Novák. 2021. An Unbiased Ray-Marching Transmittance Estimator. *ACM Trans. Graph.* 40, 4 (2021).

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980

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.

Harold J. Kushner and George Yin. 2003. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer.

Nicolas Lebbe, Charles Dapogny, Edouard Oudet, Karim Hassan, and Alain Gliere. 2019. Robust shape and topology optimization of nanophotonic devices using the level set method. *J. Comput. Phys.* 395 (2019), 710–746.

Alan J. Lee. 1990. *U-Statistics: Theory and Practice* (1st ed.). Routledge. https://doi.org/10.1201/9780203734520

Chunming Li, Chenyang Xu, Changfeng Gui, and Martin D Fox. 2005. Level set evolution without re-initialization: a new variational formulation. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR 05)*, Vol. 1. IEEE, 430–436.

Dingzeyu Li, David I. W. Levin, Wojciech Matusik, and Changxi Zheng. 2016. Acoustic voxels: computational optimization of modular acoustic filters. *ACM Trans. Graph.* 35, 4, Article 88 (jul 2016), 12 pages. https://doi.org/10.1145/2897824.2925960

Tzu-Mao Li, Miika Aittala, Fredo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Transactions on Graphics (TOG)* (2018).

Yifei Li, Tao Du, Sangeetha Grama Srinivasan, Kui Wu, Bo Zhu, Eftychios Sifakis, and Wojciech Matusik. 2022. Fluidic topology optimization with an anisotropic mixture model. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–14.

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*. 1–10.

Selena Ling, Nicholas Sharp, and Alec Jacobson. 2022. VectorAdam for Rotation Equivariant Geometry Optimization. *arXiv preprint arXiv:2205.13599* (2022).

Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-band topology optimization on a sparsely populated grid. *ACM Trans. Graph.* 37, 6, Article 251 (dec 2018), 14 pages. https://doi.org/10.1145/3272127.3275012

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38, 6 (Dec. 2019). https://doi.org/10.1145/3355089.3356510

Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. 2021. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing* 43, 6 (2021), B1105–B1132.

Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. 2021. Unified Shape and SVBRDF Recovery using Differentiable Monte Carlo Rendering. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.14344

Miles Macklin. 2022. Warp: A High-performance Python Framework for GPU Simulation and Graphics. https://github.com/nvidia/warp. NVIDIA GPU Technology Conference (GTC).

Zoë Marschner, Paul Zhang, David Palmer, and Justin Solomon. 2021. Sum-of-squares geometry processing. *ACM Trans. Graph.* 40, 6, Article 253 (dec 2021), 13 pages. https://doi.org/10.1145/3478513.3480551

Ishit Mehta, Manmohan Chandraker, and Ravi Ramamoorthi. 2022. A Level Set Theory for Neural Implicit Evolution Under Explicit Flows. In *Computer Vision – ECCV 2022*. Springer Nature Switzerland, Cham, 711–729.

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

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2024. Walkin' Robin: Walk on Stars with Robin Boundary Conditions. *ACM Trans. Graph.* 43, 4 (2024).

Mervin E Muller. 1956. Some continuous Monte Carlo methods for the Dirichlet problem. *The Annals of Mathematical Statistics* 27, 3 (1956), 569–589.

Mohammad Sina Nabizadeh, Ravi Ramamoorthi, and Albert Chern. 2021. Kelvin transformations for simulations on infinite domains. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–15.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 40, 6 (Dec. 2021). https://doi.org/10.1145/3478513.3480501

Baptiste Nicolet, Fabrice Rousselle, Jan Novak, Alexander Keller, Wenzel Jakob, and Thomas Müller. 2023. Recursive Control Variates for Inverse Rendering. *ACM Trans. Graph.* 42, 4, Article 62 (jul 2023), 13 pages. https://doi.org/10.1145/3592139

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. 2020. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3504–3515.

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 146–1.

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38, 6 (Dec. 2019). https://doi.org/10.1145/3355089.3356498

Alexandrina Orzan, Adrien Bousseau, Pascal Barla, Holger Winnemöller, Joëlle Thollot, and David Salesin. 2013. Diffusion curves: a vector representation for smooth-shaded

images. *Commun. ACM* 56, 7 (jul 2013), 101–108. https://doi.org/10.1145/2483852.2483873
Stanley Osher and Ronald P Fedkiw. 2005. *Level set methods and dynamic implicit surfaces*. Vol. 1. Springer New York.
Gustavo T Pfeiffer and Yoichi Sato. 2018. On stochastic optimization methods for Monte Carlo least-squares problems. *arXiv preprint arXiv:1804.10079* (2018).
Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically based rendering: From theory to implementation.* The MIT Press.
Olivier Pironneau. 1984. *Optimal shape design for elliptic systems* (1st ed. 1984. ed.). Springer-Verlag, New York.
Yang Qi, Dario Seyb, Benedikt Bitterli, and Wojciech Jarosz. 2022. A bidirectional formulation for Walk on Spheres. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 51–62.
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.
Dan Raviv, Christopher Barsi, Nikhil Naik, Micha Feigin, and Ramesh Raskar. 2014. Pose estimation using time-resolved inversion of diffuse light. *Opt. Express* 22, 17 (Aug 2014), 20164–20176. https://doi.org/10.1364/OE.22.020164
Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. 2020. Meshsdf: Differentiable iso-surface extraction. *Advances in Neural Information Processing Systems* 33 (2020), 22468–22478.
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.
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 (2020).
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.* (2023). 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.* (2022). https://doi.org/10.1145/3528223.3530134
Martin Schweiger, Simon R. Arridge, Mutsuhisa Hiraoka, and David T. Delpy. 1995. The finite element method for the propagation of light in scattering media: Boundary and source conditions. *Medical Physics* 22, 11 (1995), 1779–1792.
Nicholas Sharp and Keenan Crane. 2018. Variational Surface Cutting. *ACM Trans. Graph.* 37, 4 (2018).
Nicholas Sharp and Alec Jacobson. 2022. Spelunking the deep: Guaranteed queries on general neural implicit surfaces via range analysis. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–16.
Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. 2021. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems* 34 (2021), 6087–6101.
Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. 2023. Flexible Isosurface Extraction for Gradient-Based Mesh Optimization. *ACM Trans. Graph.* 42, 4, Article 37 (jul 2023), 16 pages. https://doi.org/10.1145/3592430
Jos Stam and Ryan Schmidt. 2011. On the velocity of an implicit surface. *ACM Transactions on Graphics (TOG)* 30, 3 (2011), 1–7.
Ryusuke Sugimoto, Christopher Batty, and Toshiya Hachisuka. 2024. Velocity-Based Monte Carlo Fluids. *arXiv preprint arXiv:2404.16274* (2024).
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, Article 81 (jul 2023), 16 pages. https://doi.org/10.1145/3592109
Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe control of smoke simulations. *ACM Trans. Graph.* 22, 3 (jul 2003), 716–723. https://doi.org/10.1145/882262.882337
Gunther Uhlmann. 2009. Electrical impedance tomography and Calderón's problem. *Inverse problems* 25, 12 (2009), 123011.
Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path replay backpropagation: differentiating light paths using constant memory and linear time. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.
Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable signed distance function rendering. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–18.
Lihong V. Wang and Hsin-i Wu. 2007. *Biomedical optics principles and imaging.* Wiley, Somerset, NJ, USA.
Emily Whiting, Hijung Shin, Robert Wang, John Ochsendorf, and Frédo Durand. 2012. Structural Optimization of 3D Masonry Buildings. *ACM Transactions on Graphics* 31, 6 (November 2012), 1–11. Author's final manuscript.
Zihan Yu, Lifan Wu, Zhiqian Zhou, and Shuang Zhao. 2024. A Differential Monte Carlo Solver For the Poisson Equation. *ACM Trans. Graph.* 43 (July 2024). https://doi.org/10.1145/3641519.3657460
Ekrem Fatih Yılmazer, Delio Vicini, and Wenzel Jakob. 2022. Solving Inverse PDE Problems using Grid-Free Monte Carlo Estimators. arXiv:cs.GR/2208.02114
Joseph N Zalameda and F Raymond Parker. 2014. Thermal inspection of composite honeycomb structures. In *Thermosense: Thermal Infrared Applications XXXVI*, Vol. 9105. SPIE, 66–73.
Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. 2021. Monte Carlo estimators for differential light transport. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–16.
Yong Zhan, Sanjay V. Kumar, and Sachin S. Sapatnekar. 2008. Thermally Aware Design. *Foundations and Trends® in Electronic Design Automation* 2, 3 (2008), 255–370. https://doi.org/10.1561/1000000007
Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph.* 39, 4 (2020), 143:1–143:19.
Ziyi Zhang, Nicolas Roussel, and Wenzel Jakob. 2023. Projective Sampling for Differentiable Rendering of Geometry. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–14.
Shuang Zhao, Frédo Durand, and Changxi Zheng. 2017. Inverse diffusion curves using shape optimization. *IEEE transactions on visualization and computer graphics* 24, 7 (2017), 2153–2166.
Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. 2020. Physics-Based Differentiable Rendering: From Theory to Implementation. In *ACM SIGGRAPH 2020 Courses*. https://doi.org/10.1145/3388769.3407454
Bo Zhu, Mélina Skouras, Desai Chen, and Wojciech Matusik. 2017. Two-Scale Topology Optimization with Microstructures. *ACM Trans. Graph.* 36, 4, Article 120b (jul 2017), 16 pages. https://doi.org/10.1145/3072959.3095815
ChunGang Zhuang, ZhenHua Xiong, and Han Ding. 2007. A level set method for topology optimization of heat conduction problem under multiple load cases. *Computer methods in applied mechanics and engineering* 196, 4-6 (2007), 1074–1084.

A DIFFERENTIATION OF DIRICHLET BOUNDARY CONDITION WITH TEXTURE MAPPING

We consider boundary data $g := g_M \circ \Psi$ that composes the inverse parameterization $\Psi(\cdot, \pi) : \partial\Omega(\pi) \to [0,1]^2$ with a texture function $g_M(\cdot, \pi) : [0,1]^2 \to \mathbb{R}$. We determine the boundary condition to use in the differential BVP by implicitly differentiating the boundary condition of the BVP (3)—which we reproduce here:

$$u(x,\pi) = g(x,\pi) = g_M(\Psi(x,\pi),\pi) \quad \text{on } \partial\Omega(\pi). \tag{26}$$

Differentiating the left-hand side and using the chain rule yields:

$$\frac{\mathrm{d}}{\mathrm{d}\pi} u(x,\pi) = \dot{u}(x,\pi) + \nabla u(x,\pi) \cdot \mathrm{V}(x,\pi). \tag{27}$$

Doing likewise for the right-hand side gives us:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\pi} g_M(\Psi(x,\pi),\pi) &= \dot{g}_M(\Psi(x,\pi),\pi) \\ &\quad + \nabla g_M(\Psi(x,\pi),\pi) \frac{\mathrm{d}}{\mathrm{d}\pi}\Psi(x,\pi) \qquad (28)\\ &= \dot{g}_M(\Psi(x,\pi),\pi) \\ &\quad + \nabla g_M(\Psi(x,\pi),\pi) \\ &\qquad \cdot \left(\dot{\Psi}(x,\pi) + \nabla\Psi(x,\pi)\,\mathrm{V}(x,\pi)\right). \qquad (29)\end{aligned}$$

We use the identity $\nabla g_M \cdot \nabla\Psi = \nabla g$ and make dependence on π implicit to simplify this expression:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\pi} g_M(\Psi(x)) &= \dot{g}_M(\Psi(x)) \\ &\quad + \nabla g_M(\Psi(x)) \cdot \dot{\Psi}(x) + \nabla g(x)\,\mathrm{V}(x,\pi). \qquad (30)\end{aligned}$$

Lastly, we note that, as Equation (26) requires that u and g agree along the boundary, their tangential gradients are the same. Thus,

by equating Equations (27) and (30) and eliminating tangential components on both sides, we arrive at the boundary condition:

$$\dot{u}(x) = \dot{\mathrm{g}}_M(\Psi(x)) + \nabla \mathrm{g}_M(\Psi(x)) \cdot \dot{\Psi}(x) + \left(\frac{\partial \mathrm{g}}{\partial n}(x) - \frac{\partial u}{\partial n}(x)\right) \mathrm{V}_n(x,\pi) \quad \text{on } \partial\Omega(\pi), \tag{31}$$

which is the same as Equation (10).

B THE ADJOINT BOUNDARY VALUE PROBLEM

We briefly comment on an alternative formulation for computing derivatives of PDE-constrained shape functionals $\mathcal{L}(\pi)$ as in Equation (6). Zhao et al. [2017] and Henrot and Pierre [2018, Section 5.8] show that we can express the derivative of the shape functional as:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\pi}(\pi) = \int_{\partial\Omega(\pi)} \frac{\partial a}{\partial n}(y)\left(\frac{\partial \mathrm{g}}{\partial n}(y) - \frac{\partial u}{\partial n}(y)\right) \mathrm{V}_n(y,\pi)\, \mathrm{d}\sigma(y), \tag{32}$$

where $a(\cdot,\pi) : \Omega(\pi) \to \mathbb{R}$ is the solution to the *adjoint boundary value problem*:

$$\begin{aligned} \Delta a(x) - \sigma a(x) &= \mathrm{L}'(u(x)) && \text{in } \Omega(\pi), \\ a(x) &= 0 && \text{on } \partial\Omega(\pi). \end{aligned} \tag{33}$$

Compared to Equation (7), Equation (32) is simpler because it requires estimating only a boundary integral and no domain integral. Compared to the differential BVP (11), the adjoint BVP is also a screened Poisson equation with Dirichlet boundary conditions; however, the primary BVP is nested within the source term, and not the boundary data. Consequently, though the solution a could still be estimated using WoS, nesting is more difficult: As the source term is invoked at every recursion step of WoS, nesting would require launching a walk for estimating u at *every step* along the walk for estimating a, resulting in quadratic complexity. By contrast, nesting for the differential BVP requires launching a walk for estimating u at *only the last step* of the walk for estimating $\dot{u}$, maintaining linear complexity (Algorithm 2). This complexity difference motivated our choice to use the differential BVP. The quadratic complexity of the adjoint approach could potentially be overcome using off-centered connections ([Sawhney et al. 2022, Section 5.2] and [Sawhney and Crane 2020, Section 4.3]) to merge the walks for u back into the walk for a; or caching schemes [Miller et al. 2023; Li et al. 2023] that estimate u without a walk. Such approaches are exciting future research directions.

C MORE GENERAL SHAPE FUNCTIONAL

In some of our experiments in Section 7, we use a more general shape functional of the form:

$$\mathcal{L}(\pi) \coloneqq \int_{\Omega(\pi)} \mathrm{M}(x)\, \mathrm{L}(u(x,\pi))\, \mathrm{d}x + \int_{\partial\Omega(\pi)} \mathrm{m}(y)\, \mathrm{l}(u(y,\pi))\, \mathrm{dA}(y), \tag{34}$$

where $\mathrm{L}, \mathrm{l} : \mathbb{R} \to \mathbb{R}$ are differentiable loss functions, and $\mathrm{M}, \mathrm{m} : \mathbb{R}^3 \to \{0,1\}$ are binary mask functions. Compared to Equation (6), the shape functional of Equation (34) includes an additional loss term evaluated on the boundary $\partial\Omega(\pi)$. Differentiating this shape functional produces [Henrot and Pierre 2018, p. 239]:

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\pi}(\pi) = &\int_{\Omega(\pi)} \mathrm{M}(x)\, \dot{u}(x,\pi)\, \mathrm{L}'(u(x,\pi))\, \mathrm{d}x \\ &+ \int_{\partial\Omega(\pi)} \mathrm{m}(y)\, \dot{u}(y,\pi)\, \mathrm{l}'(u(y,\pi)) \\ &\quad + \mathrm{V}_n(y,\pi)\Big(\mathrm{M}(y)\, \mathrm{L}(u(y,\pi)) \\ &\qquad + \mathrm{m}(y)\, \kappa(y)\, \mathrm{l}(u(y,\pi)) \\ &\qquad + \mathrm{m}(y)\, \mathrm{l}'(u(y,\pi)) \frac{\partial u}{\partial n}(y,\pi)\Big)\, \mathrm{dA}(y), \end{aligned} \tag{35}$$

where L', l' are the derivatives of the scalar losses L, l; and κ is the mean curvature. Equation (35) is valid under the same two assumptions we made for Equation (7), and requires computing values of u, $\partial u/\partial n$, and $\dot{u}$ like that equation. Thus we can estimate the shape functional derivative of Equation (35) using the same algorithm we developed in Section 5 for the derivative of Equation (7).