A Differential Monte Carlo Solver For the Poisson Equation

Zihan Yu
zihay19@uci.edu
University of California, Irvine & NVIDIA
USA

Lifan Wu
lifanw@nvidia.com
NVIDIA
USA

Zhiqian Zhou
zhiqiaz8@uci.edu
University of California, Irvine
USA

Shuang Zhao
shz@ics.uci.edu
University of California, Irvine & NVIDIA
USA

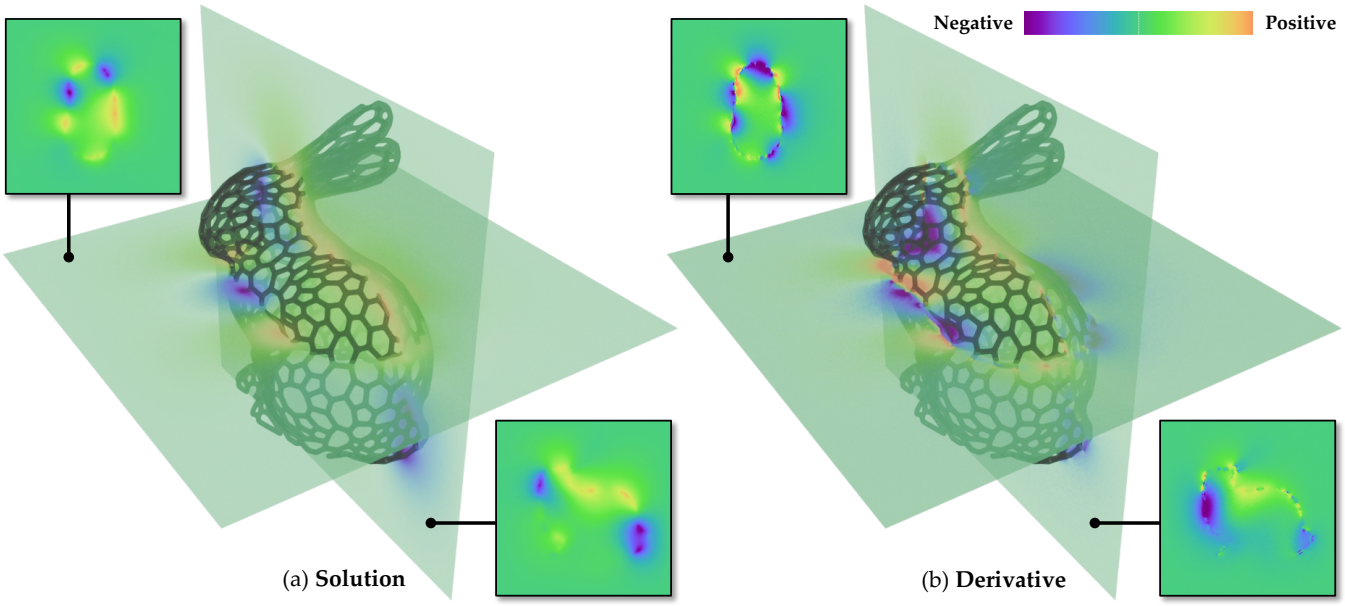

Figure 1: We introduce a new grid-free technique to estimate derivatives of solutions to the Poisson equation with respect to arbitrary parameters including domain shapes. This example includes a 3D Laplace problem with Dirichlet boundary conditions on a wired bunny shape. We visualize the solution to this problem in two cross-sectional planes in (a) and the derivative of this solution (with respect to the translation of the bunny) estimated with our method in (b).

ABSTRACT

The Poisson equation is an important partial differential equation (PDE) with numerous applications in physics, engineering, and computer graphics. Conventional solutions to the Poisson equation require discretizing the domain or its boundary, which can be very expensive for domains with detailed geometries. To overcome this challenge, a family of grid-free Monte Carlo solutions has recently been developed. By utilizing walk-on-sphere (WoS) processes, these techniques are capable of efficiently solving the Poisson equation over complex domains.

In this paper, we introduce a general technique that differentiates solutions to the Poisson equation with Dirichlet boundary conditions. Specifically, we devise a new boundary-integral formulation for the derivatives with respect to arbitrary parameters including shapes of the domain. Further, we develop an efficient walk-on-spheres technique based on our new formulation—including a new approach to estimate normal derivatives of the solution field. We demonstrate the effectiveness of our technique over baseline methods using several synthetic examples.

SIGGRAPH Conference Papers '24, July 27–August 01, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657460

CCS CONCEPTS

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

KEYWORDS

Monte Carlo methods, differentiation, walk on spheres

ACM Reference Format:

Zihan Yu, Lifan Wu, Zhiqian Zhou, and Shuang Zhao. 2024. A Differential Monte Carlo Solver For the Poisson Equation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641519.3657460

1 INTRODUCTION

The Poisson equation is an elliptic partial differential equation (PDE) commonly used to model various phenomena including heat conduction, fluid dynamics, and electrostatics and, thus, has numerous applications in physics, engineering, and computer graphics.

Conventionally, the Poisson equation is usually solved using numerical methods such as the finite element method (FEM) that first discretizes the problem and then solves the resulting system of linear equations. Unfortunately, for problems over complex domains, the discretization can become extremely expensive, significantly limiting the practicality of these discretization-based methods.

Recently, a family of grid-free Monte Carlo methods [Sawhney and Crane 2020; Sawhney et al. 2022; Qi et al. 2022; Sawhney et al. 2023] utilizing the walk-on-spheres (WoS) process [Muller 1956] have been introduced to solve second-order linear elliptical PDEs. These techniques do not require discretizing domains or boundaries, allowing them to scale well to problems with complex domain geometries.

In this paper, we focus on the problem of differentiating solutions to the Poisson equation with respect to arbitrary parameters—including the shape of the domain. Techniques capable of estimating these derivatives efficiently will be a key ingredient for solving *inverse Poisson problems*—which involve inferring the source, the boundary condition, or domain shape of a Poisson problem given the solution scalar field.

Previously, derivatives of PDE solutions have mostly been estimated using discretization-based methods (by applying automatic differentiation to both the discretization and the linear solve steps) and, hence, have difficulties scaling to domains with complex geometries.

To address this problem, we introduce a grid-free Monte Carlo technique utilizing the walk-on-spheres (WoS) process. Our method offers the efficiency to handle domains with complex geometries and the generality of differentiating with respect to arbitrary parameters including the shape of the domain.

Concretely, our contributions include:

- Devising a new *boundary*-integral formulation for derivatives of solutions to the Poisson equation with respect to arbitrary parameters (§4).
- Developing a new Monte Carlo algorithm based on our new formulation (§5). A key component of our algorithm is a new walk-on-spheres (WoS) estimator for normal derivatives of the solution field over the domain boundary (§5.2).

We evaluate our technique empirically in §6 using several synthetic examples.

2 RELATED WORKS

Grid-free Monte Carlo PDE solvers. Sawhney and Crane [2020] have introduced to graphics a *grid-free* Monte Carlo technique for solving Poisson equations with Dirichlet boundary conditions by leveraging the walk-on-spheres (WoS) process [Muller 1956]. This technique does not require discretizing space and, therefore, scales well to problems over complex domains. Later, this technique has been generalized to handle second-order linear elliptic PDEs (e.g., screened Poisson equations) [Sawhney et al. 2022] and Neumann boundary conditions [Sawhney et al. 2023].

Further, several techniques have been developed to speed up the convergence of WoS solvers. Qi et al. [2022], for instance, have proposed a bidirectional sampling method capable of handling problems with concentrated sources. Miller et al. [2023] have introduced a caching scheme to allow path samples generated by WoS to be shared among multiple queries.

As an alternative to WoS, Sugimoto et al. [2023] have developed a technique based on the walk-on-boundary (WoB) process [Sabelfeld 1982]. Compared with the WoS-based methods, this technique usually offers better performance over (mostly) convex domains.

Differentiable grid-free solvers. Despite the great progress made for "forward" solvers, differentiable grid-free PDE solvers have remained lacking. Recently, Yilmazer et al. [2022] have differentiated the grid-free solver by Sawhney et al. [2022] over *fixed* domains. Additionally, for solving inverse diffusion curve [Orzan et al. 2008] problems, Zhao et al. [2018] have derived shape derivatives of solutions to Laplace's equation—a special case of the Poisson equation with zero source. Numerically, they relied on conventional finite-element method (FEM) to compute the derivatives.

Our mathematical formulation—which we will present in §4—is a significant generalization of these works.

Shape optimization. As a subfield of optimal control theory, shape optimization [Sokolowski and Zolésio 1992; Haslinger and Mäkinen 2003; Delfour and Zolsio 2010; Walker 2015] tackles the problem of finding a bounded domain Ω to minimize a continuous functional on Ω. Our theory (§4) enables solving shape optimizations with Poisson-PDE constraints—which is significantly more challenging than conventional shape optimization problems. Additionally, our grid-free Monte Carlo estimators (§5) allow the optimization of highly detailed shapes.

3 PRELIMINARIES

In general, the Poisson equation (with Dirichlet boundary condition) takes the form:

$$\begin{aligned} \Delta u &= -f \quad \text{on } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where Ω is an open subset of $\mathbb{R}^n$ with (closed) boundary $\partial\Omega$.[1] Additionally, $f : \Omega \mapsto \mathbb{R}$ and $g : \partial\Omega \mapsto \mathbb{R}$ are, respectively, the **source** and **boundary** functions.

Representation formula. It has been shown that the solution u of the Poisson equation (1) can be expressed using the **representation**

[1] In this paper, we focus on 2D and 3D domains (i.e., $n = 2, 3$).

formula [Evans 2010] as

$$u(\boldsymbol{x}) = \underbrace{\int_\Omega f(\boldsymbol{y})\, \mathcal{G}^\Omega(\boldsymbol{x} \leftrightarrow \boldsymbol{y})\, \mathrm{d}\boldsymbol{y}}_{\text{interior}} + \underbrace{\int_{\partial\Omega} g(z)\, \mathcal{P}^\Omega(\boldsymbol{x} \to z)\, \mathrm{d}z}_{\text{boundary}}. \quad (2)$$

In this equation, $\mathcal{G}^\Omega$ is the (domain-specific) **Green's function** satisfying that, for all $\boldsymbol{x}, \boldsymbol{y} \in \Omega$, $\mathcal{G}^\Omega(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) = u'(\boldsymbol{y})$ where u' is the solution of a Poisson problem with delta sources:

$$\begin{aligned} \Delta u' &= -\delta_{\boldsymbol{y}}(\boldsymbol{x}) \quad && \text{on } \Omega, \\ u' &= 0 && \text{on } \partial\Omega, \end{aligned} \quad (3)$$

with $\delta_{\boldsymbol{y}}$ being the impulse function centered at $\boldsymbol{y}$.

In addition, $\mathcal{P}^\Omega$ is the **Poisson kernel** associated with the domain Ω. For all $\boldsymbol{x} \in \Omega$ and $z \in \partial\Omega$, it holds that

$$\mathcal{P}^\Omega(\boldsymbol{x} \to z) = \partial_{\boldsymbol{n}(z)} \mathcal{G}^\Omega(\boldsymbol{x} \leftrightarrow z) = \boldsymbol{n}(z) \cdot \frac{\partial}{\partial z} \mathcal{G}^\Omega(\boldsymbol{x} \leftrightarrow z), \quad (4)$$

where $\boldsymbol{n}$ indicates the inward unit-normal field of the boundary $\partial\Omega$, $\partial_{\boldsymbol{n}(z)} \mathcal{G}^\Omega(\boldsymbol{x} \leftrightarrow z)$ is the **normal derivative** of $\mathcal{G}^\Omega(\boldsymbol{x} \leftrightarrow z)$ at z (with $\boldsymbol{x}$ fixed), and "$\cdot$" denotes the dot-product operator.

Except for domains with very simple shapes (e.g., spheres), the Green's function $\mathcal{G}^\Omega$ and Poisson kernel $\mathcal{P}^\Omega$ generally have no analytical expressions.

Integral-equation formulation. Besides Eq. (2), it is also possible to write the solution u as the solution of an integral equation [Evans 2010]:

$$u(\boldsymbol{x}) = \underbrace{\int_{B_{\boldsymbol{x}}} f(\boldsymbol{y})\, G(\boldsymbol{x} \leftrightarrow \boldsymbol{y})\, \mathrm{d}\boldsymbol{y}}_{\text{interior}} + \underbrace{\int_{\partial B_{\boldsymbol{x}}} u(z)\, P(\boldsymbol{x} \to z)\, \mathrm{d}z}_{\text{boundary}}. \quad (5)$$

where:

- $B_{\boldsymbol{x}} \subseteq \Omega \cup \partial\Omega$ denotes a ball centered at $\boldsymbol{x}$;
- G and P are, respectively, the Green's function and Poisson kernel associated with the ball $B_{\boldsymbol{x}}$. Please see the work by Sawhney and Crane [2020] for the analytical expressions of these functions.

Walk on spheres. The integral equation of Eq. (5) can be solved numerically using an algorithm called *walk on spheres*—which we briefly describe in the following.

Let $\Omega_\epsilon \subset \Omega$ be a ϵ-**shell** comprised of all points $\boldsymbol{x} \in \Omega$ no more than ϵ distance away from the domain boundary $\partial\Omega$ (for some small $\epsilon \in \mathbb{R}_+$). Then, the solution $u(\boldsymbol{x})$ of the Poisson equation (1) can be approximated using the Monte Carlo estimator introduced to graphics by Sawhney and Crane [2020]:

$$\langle u_\epsilon(\boldsymbol{x}) \rangle_{\text{wos}} = \begin{cases} g(\boldsymbol{x}_\perp), & (\boldsymbol{x} \in \Omega_\epsilon) \\ \frac{f(\boldsymbol{y})\, G(\boldsymbol{x} \leftrightarrow \boldsymbol{y})}{p(\boldsymbol{y})} + \frac{\langle u_\epsilon(z) \rangle_{\text{wos}}\, P(\boldsymbol{x} \to z)}{p(z)}, & (\boldsymbol{x} \notin \Omega_\epsilon) \end{cases} \quad (6)$$

where:

- $\boldsymbol{x}_\perp$ is the projection of $\boldsymbol{x}$ on $\partial\Omega$;
- The points $\boldsymbol{y}$ and z are drawn, respectively, from the ball $B_{\boldsymbol{x}}$ and its boundary $\partial B_{\boldsymbol{x}}$ with the probability densities $p(\boldsymbol{y})$ and $p(z)$;
- $\langle u_\epsilon(z) \rangle_{\text{wos}}$ is estimated recursively.

Table 1: List of symbols commonly used in this paper.

Symbol	Definition
u	solution to the Poisson equation (1)
f	source function of the Poisson equation (1)
g	boundary function of the Poisson equation (1)
Ω, $\partial\Omega$	evolving domain and its boundary
$\hat{\Omega}$, $\partial\hat{\Omega}$	reference domain and its boundary
$\mathsf{X}(\cdot, \theta)$	one-to-one mapping from $\hat{\Omega}$ to $\Omega(\theta)$
$\mathsf{X}^{-1}(\cdot, \theta)$	inverse of $\mathsf{X}(\cdot, \theta)$, transforms $\Omega(\theta)$ to $\hat{\Omega}$
X^*	pull-back operator
$\hat{u}$	pull-back of the solution u (i.e., $\hat{u} := \mathsf{X}^* u$)
$\hat{g}$	pull-back of the boundary function g (i.e., $\hat{g} := \mathsf{X}^* g$)
$\mathcal{G}^\Omega$, $\mathcal{G}^{\hat{\Omega}}$	Green's functions associated with Ω and $\hat{\Omega}$
$\mathcal{P}^\Omega$, $\mathcal{P}^{\hat{\Omega}}$	Poisson kernels associated with Ω and $\hat{\Omega}$
∂_θ	derivative wrt. θ at $\theta = 0$ defined in Eq. (7)
$\boldsymbol{v}$	"velocity" of a point defined in Eq. (12)
Ω_ϵ, $\hat{\Omega}_\epsilon$	ϵ-shells of the evolving Ω and the reference domain $\hat{\Omega}$
$B_{\boldsymbol{x}}$, $B_{\boldsymbol{c}}$	balls centered at $\boldsymbol{x}$ and $\boldsymbol{c}$

Prior works (e.g., [Muller 1956; Delaurentis and Romero 1990]) have shown that (the expected value of) Eq. (6) closely approximates the true solution u when ϵ is near zero.

Additionally, the estimator outlined in Eq. (6) can be further improved by using, for example, bidirectional sampling [Qi et al. 2022] or caching [Miller et al. 2023]. We opt for the "basic" version in this paper since these improvements are orthogonal to our technique.

4 BOUNDARY-INTEGRAL FORMULATION

We now derive the derivative $\partial u / \partial\theta$ of the solution u to the Poisson equation (1) with respect to some parameter $\theta \in \mathbb{R}$. Without loss of generality, we assume the derivative to be evaluated to be at $\theta = 0$. In other words, letting

$$\partial_\theta h := \left[\frac{\partial h}{\partial \theta} \right]_{\theta=0}, \quad \text{for any } h, \quad (7)$$

our goal is to obtain $(\partial_\theta u)(\boldsymbol{x})$ for any fixed $\boldsymbol{x} \in \Omega(0)$.

Simple case. When the parameter θ controls the source and boundary functions (i.e., f and g) only (but not the domain Ω), the derivative $\partial_\theta u$ is essentially the solution of another Poisson problem with the source and boundary functions differentiated:

$$\begin{aligned} \Delta(\partial_\theta u) &= -\partial_\theta f \quad && \text{on } \Omega, \\ \partial_\theta u &= \partial_\theta g && \text{on } \partial\Omega. \end{aligned} \quad (8)$$

As demonstrated by Yilmazer et al. [2022], Eq. (8) can be solved using the same walk-on-spheres method outlined in Eq. (6).

General case. When the domain Ω evolves with the parameter θ, on the other hand, the evaluation of the derivative $\partial_\theta u$ becomes more challenging. To this end, one solution is to differentiate the integral equation (5) using Reynolds transport theorem [1903]. Unfortunately, doing so requires handling discontinuities emerging from the dependencies of the ball $B_{\boldsymbol{x}}$ and the ϵ-shell Ω_ϵ on θ.

We introduce a new formulation for the derivative $\partial_\theta u$. As we will demonstrate in §5, our formulation allows the development of significantly more efficient Monte Carlo estimators.

In what follows, we present the material-form parameterization of the domain in §4.1 before introducing our derivation of the derivative $\partial_\theta u$ in §4.2.

We summarize the commonly used symbols in Table 1.

4.1 Material-Form Parameterization

To facilitate the differentiation of the solution u, we first parameterize the evolving domain $\Omega(\theta)$ using a fixed one. This parameterization is known as the *material* or *Lagrangian* form.

Let

$$\hat{\Omega} \coloneqq \Omega(0), \tag{9}$$

be a **reference domain** that is independent of the parameter θ. Then, the evolution of the domain $\Omega(\theta)$ can be expressed using a one-to-one mapping $\mathsf{X}(\cdot, \theta)$ that, for any θ, transforms the reference domain $\hat{\Omega}$ to the evolving $\Omega(\theta)$ and the boundary $\partial\hat{\Omega}$ to $\partial\Omega(\theta)$. We call any $\boldsymbol{x} \in \Omega(\theta)$ a **spatial point** and $\boldsymbol{p} \in \hat{\Omega}$ a **material point**.

The pull-back operator. Any scalar field h over the evolving domain $\Omega(\theta)$ can be transformed to another scalar field $\mathsf{X}^* h$ over the reference domain $\hat{\Omega}$ using the **pull-back operator** X^* such that That is,

$$(\mathsf{X}^* h)(\boldsymbol{p}) \coloneqq h(\mathsf{X}(\boldsymbol{p}, \theta)), \quad \text{for any } \boldsymbol{p} \in \hat{\Omega}. \tag{10}$$

Special case. At $\theta = 0$, according to Eq. (9), the evolving domain Ω coincides with the reference one $\hat{\Omega}$. In this case, we assume the mapping $\mathsf{X}(\cdot, 0)$ and its inverse $\mathsf{X}^{-1}(\cdot, 0)$ to both reduce to identity maps. It follows that the induced pull-back operator also reduces to identity.

4.2 Our Derivation

We now derive the derivative of u based on the material-form parameterization presented in §4.1.

Assumptions. To ensure the continuity of the solution u, we assume that (i) the source f and boundary functions g are both C^0-continuous and differentiable; and (ii) the domain boundary $\partial\Omega$ is G^1 (with continuous surface normal).

Additionally, to simplify derivations, we assume that (iii) the source f and boundary function g are defined over the entirety of the ambient space (e.g., $\mathbb{R}^2$ or $\mathbb{R}^3$) in which the domain $\Omega(\theta)$ evolves, and (iv) f is independent of the parameter θ. As discussed in the supplement, assumptions (iii) and (iv) can be relaxed easily.

Derivation outline. For any spatial point $\boldsymbol{x}$ satisfying $\boldsymbol{x} = \mathsf{X}(\boldsymbol{p}, \theta)$ for some fixed material point $\boldsymbol{p}$ and all $\theta \in \mathbb{R}$, the **material derivative** $\left[\frac{\mathrm{d}}{\mathrm{d}\theta} u(\boldsymbol{x})\right]_{\theta=0}$ follows the chain rule:

$$\left[\frac{\mathrm{d}}{\mathrm{d}\theta} u(\boldsymbol{x})\right]_{\theta=0} = (\partial_\theta u)(\boldsymbol{x}) + \boldsymbol{v}(\boldsymbol{p}) \cdot \nabla u(\boldsymbol{x}), \tag{11}$$

where

$$\boldsymbol{v}(\boldsymbol{p}) \coloneqq \left[\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\theta}\right]_{\theta=0} = \partial_\theta \mathsf{X}(\boldsymbol{p}, \theta), \tag{12}$$

captures the change rate (or "velocity") of $\boldsymbol{x}$ with respect to θ.

We note that, on the left-hand side of Eq. (11), the dependency of $\boldsymbol{x}$ on the parameter θ is considered when taking the derivative (which is further demonstrated in Eq. (13) below). On the other hand, the term $(\partial_\theta u)(\boldsymbol{x})$ of our interest on the right-hand side considers only the dependency of u on θ.

Based on Eq. (11), we derive the partial derivative $(\partial_\theta u)(\boldsymbol{x})$ in the following by first obtaining the material derivative $\left[\frac{\mathrm{d}}{\mathrm{d}\theta} u(\boldsymbol{x})\right]_{\theta=0}$ and then subtracting $\boldsymbol{v} \cdot \nabla u$.

The material derivative. Let $\hat{u} \coloneqq \mathsf{X}^* u$ be the solution u of the Poisson equation (1) mapped to the reference domain using the pull-back operator X^*. Given any spatial point $\boldsymbol{x} = \mathsf{X}(\boldsymbol{p}, \theta)$ for some fixed material point $\boldsymbol{p}$ and all $\theta \in \mathbb{R}$, the material derivative $\left[\frac{\mathrm{d}}{\mathrm{d}\theta} u(\boldsymbol{x})\right]_{\theta=0}$ satisfies that

$$\begin{aligned}\left[\frac{\mathrm{d}}{\mathrm{d}\theta} u(\boldsymbol{x})\right]_{\theta=0} &= \lim_{\epsilon \to 0} \frac{u_\epsilon(\mathsf{X}(\boldsymbol{p}, \epsilon)) - u_0(\mathsf{X}(\boldsymbol{p}, 0))}{\epsilon} \\ &= \lim_{\epsilon \to 0} \frac{\hat{u}_\epsilon(\boldsymbol{p}) - \hat{u}_0(\boldsymbol{p})}{\epsilon} =: (\partial_\theta \hat{u})(\boldsymbol{p}),\end{aligned} \tag{13}$$

where: u_ϵ and u_0 indicate, respectively, the solutions of the Poisson equation (1) when $\theta = \epsilon$ and $\theta = 0$; and $\hat{u}_\epsilon \coloneqq \mathsf{X}^* u_\epsilon$ and $\hat{u}_0 \coloneqq \mathsf{X}^* u_0$.

According to Eq. (13), the problem of evaluating the material derivative $\left[\frac{\mathrm{d}}{\mathrm{d}\theta} u(\boldsymbol{x})\right]_{\theta=0}$ amounts to evaluating the derivative $(\partial_\theta \hat{u})(\boldsymbol{p})$, which we derive in the following.

Evaluating $\partial_\theta \hat{u}$. When $\theta = 0$, it can be shown (see page 23 of the book by Henry [2005]) that

$$\partial_\theta \left(\mathsf{X}^*(\Delta u)\right) = \boldsymbol{v} \cdot \nabla_{\hat{\Omega}}(\Delta_{\hat{\Omega}} \hat{u}) + \Delta_{\hat{\Omega}} \left(\partial_\theta \hat{u} - \boldsymbol{v} \cdot \nabla_{\hat{\Omega}} \hat{u}\right), \tag{14}$$

where $\boldsymbol{v}(\boldsymbol{p})$ is defined in Eq. (12), and $\nabla_{\hat{\Omega}}$ and $\Delta_{\hat{\Omega}}$ denote, respectively, the gradient and Laplacian operators over the reference domain $\hat{\Omega}$.

As discussed in §4.1, at $\theta = 0$ the reference domain $\hat{\Omega}$ coincides with the evolving one $\Omega(\theta)$, causing the mapping X and the induced pull-back operator X^* to both reduce to identity. This allows the left-hand side of Eq. (14) to become

$$\partial_\theta \left(\mathsf{X}^*(\Delta u)\right) = \partial_\theta (\mathsf{X}^*(-f)) = -\mathsf{X}^*(\partial_\theta f) - \boldsymbol{v} \cdot \nabla f = -\boldsymbol{v} \cdot \nabla f, \tag{15}$$

where the last equality follows the assumption that the source function f is independent of θ (i.e., $\partial_\theta f = 0$).

Additionally, since the solution u and its pull-back $\hat{u}$ coincide when $\theta = 0$, we have $\nabla_{\hat{\Omega}}(\Delta_{\hat{\Omega}} \hat{u}) = \nabla(\Delta u) = -\nabla f$. It follows that the right-hand side of Eq. (14) becomes

$$-\boldsymbol{v} \cdot \nabla f + \Delta_{\hat{\Omega}} \left(\partial_\theta \hat{u} - \boldsymbol{v} \cdot \nabla_{\hat{\Omega}} \hat{u}\right). \tag{16}$$

Given Eqs. (15) and (16), we have

$$\boxed{\Delta_{\hat{\Omega}}(\partial_\theta \hat{u}) = \Delta_{\hat{\Omega}} \left(\boldsymbol{v} \cdot \nabla_{\hat{\Omega}} \hat{u}\right),} \tag{17}$$

in the interior of the reference domain $\hat{\Omega}$.

Let $\hat{g} \coloneqq \mathsf{X}^* g$ be the pull-back of the boundary function g. Then, by applying pull-back X^* and differentiation ∂_θ to both sides of the Dirichlet boundary condition (i.e., $u = g$) of the original Poisson equation (1), we have

$$\partial_\theta \hat{u} = \partial_\theta \hat{g} \quad \text{on } \partial\hat{\Omega}. \tag{18}$$

Eqs. (17) and (18) define another Poisson problem over the fixed reference domain $\hat{\Omega}$ with the solution $\partial_\theta \hat{u}$. Then, according to the

representation formula (2), we have

$$(\partial_\theta \hat{u})(\boldsymbol{p}) = \underbrace{-\int_{\hat{\Omega}} \mathcal{G}^{\hat{\Omega}}(\boldsymbol{p} \leftrightarrow \boldsymbol{q}) \left(\Delta_{\hat{\Omega}} \left(v \cdot \nabla_{\hat{\Omega}} \hat{u}\right)\right)(\boldsymbol{q})\, \mathrm{d}\boldsymbol{q}}_{\text{interior}} + \underbrace{\int_{\partial\hat{\Omega}} \mathcal{P}^{\hat{\Omega}}(\boldsymbol{p} \to \boldsymbol{s})\, \partial_\theta \hat{g}(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}}_{\text{boundary}}, \tag{19}$$

where $\mathcal{G}^{\hat{\Omega}}$ and $\mathcal{P}^{\hat{\Omega}}$ denote the Green's function and the Poisson kernel associated with the reference domain $\hat{\Omega}$, respectively.

Completing the derivation. As shown in §1 of the supplemental document, the *interior* term of Eq. (19) further satisfies

$$\begin{aligned} &-\int_{\hat{\Omega}} \mathcal{G}^{\hat{\Omega}}(\boldsymbol{p} \leftrightarrow \boldsymbol{q}) \left(\Delta_{\hat{\Omega}} \left(v \cdot \nabla_{\hat{\Omega}} \hat{u}\right)\right)(\boldsymbol{q})\, \mathrm{d}\boldsymbol{q} \\ &= -\int_{\partial\hat{\Omega}} \mathcal{P}^{\hat{\Omega}}(\boldsymbol{p} \to \boldsymbol{s}) \left(v(\boldsymbol{s}) \cdot (\nabla_{\hat{\Omega}} \hat{u})(\boldsymbol{s})\right) \mathrm{d}\boldsymbol{s} + v(\boldsymbol{p}) \cdot (\nabla_{\hat{\Omega}} \hat{u})(\boldsymbol{p}). \end{aligned} \tag{20}$$

Provided Eqs. (19, 20) and the fact by combining Eqs. (11, 13) that

$$(\partial_\theta u)(\boldsymbol{x}) = (\partial_\theta \hat{u})(\boldsymbol{p}) - v(\boldsymbol{p}) \cdot \underbrace{\nabla u(\boldsymbol{x})}_{=\nabla_{\hat{\Omega}} \hat{u}(\boldsymbol{p})}, \quad \text{when } \theta = 0, \tag{21}$$

we rewrite the derivative $\partial_\theta u$ as one *boundary* integral:

$$\boxed{(\partial_\theta u)(\boldsymbol{p}) = \int_{\partial\hat{\Omega}} \mathcal{P}^{\hat{\Omega}}(\boldsymbol{p} \to \boldsymbol{s}) \left(\partial_\theta \hat{g}(\boldsymbol{s}) - v(\boldsymbol{s}) \cdot \nabla_{\hat{\Omega}} \hat{u}(\boldsymbol{s})\right) \mathrm{d}\boldsymbol{s},} \tag{22}$$

for any $\boldsymbol{p} \in \hat{\Omega} = \Omega(0)$.

5 DIFFERENTIAL WALK-ON-SPHERES

We now introduce a new Monte Carlo estimator for the *boundary* integral of Eq. (22). Our technique involves three main steps (see Figure 2):

S.1 Drawing a material point $\boldsymbol{s}$ from the domain boundary $\partial\hat{\Omega}$ with a probability density that equals $\mathcal{P}^{\hat{\Omega}}(\boldsymbol{p} \to \boldsymbol{s})$ using a walk-on-spheres process.

S.2 With the point $\boldsymbol{s}$ determined, estimating the gradient $\nabla_{\hat{\Omega}} \hat{u}(\boldsymbol{s})$ using another walk-on-spheres process.

S.3 Returning $\partial_\theta \hat{g}(\boldsymbol{s}) - v(\boldsymbol{s}) \cdot \nabla_{\hat{\Omega}} \hat{u}(\boldsymbol{s})$ with the terms $\partial_\theta \hat{g}(\boldsymbol{s})$ and $v(\boldsymbol{s})$ computed analytically using automatic differentiation.

In what follows, we discuss the first two steps (**S.1** and **S.2**) in detail.

5.1 Sampling Boundary Point

The first step (**S.1**) of our method involves sampling a material point $\boldsymbol{s}$ from the boundary $\partial\hat{\Omega}$ of the reference domain $\hat{\Omega}$ with the probability density $\mathcal{P}^{\hat{\Omega}}(\boldsymbol{p} \to \boldsymbol{s})$ with $\boldsymbol{p}$ fixed. Unfortunately, as discussed in §3, the Poisson kernel $\mathcal{P}^{\hat{\Omega}}$ for an arbitrary domain $\hat{\Omega}$ cannot be evaluated, let alone sampled, analytically.

To address this problem, we utilize a walk-on-sphere algorithm. As outlined in Algorithm 1, this process essentially simulates a particle $\boldsymbol{q} \in \hat{\Omega}$ that undergoes a Brownian motion. Starting from the point $\boldsymbol{p}$, the process returns the position of $\boldsymbol{q}$ when it first reaches the boundary $\partial\hat{\Omega}$.

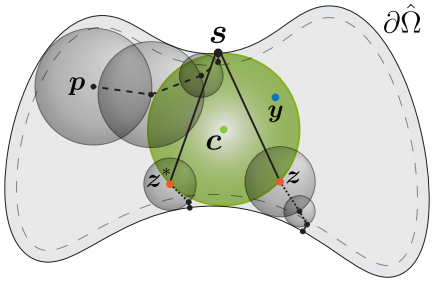

Figure 2: Our estimator: To estimate $(\partial_\theta \hat{u})(\boldsymbol{p})$ for some given $\boldsymbol{p} \in \hat{\Omega}$, our method utilizes two walk-on-spheres (WoS) processes. The first WoS (illustrated as dashed lines) constructs a path from $\boldsymbol{p}$ to a random surface point $\boldsymbol{s}$ with the probability $\mathcal{P}^{\hat{\Omega}}(\boldsymbol{p} \to \boldsymbol{s})$. Then, to estimate the normal derivative $\partial_{\boldsymbol{n}(\boldsymbol{s})} u$, we find the largest ball $B_{\boldsymbol{c}} \subseteq \hat{\Omega}$ (shown in green) tangent to the boundary $\partial\hat{\Omega}$ at the point $\boldsymbol{s}$. With the ball $B_{\boldsymbol{c}}$ determined, we antithetically sample a pair of points z and z^* on the boundary $\partial B_{\boldsymbol{c}}$ of the ball and start the second WoS (shown as dotted lines) from these locations to estimate the solutions $u(z)$ and $u(z^*)$. In addition, we draw a point $\boldsymbol{y}$ inside the ball $B_{\boldsymbol{c}}$ to evaluate contributions of the source function.

ALGORITHM 1: Sampling a material point $\boldsymbol{s} \in \partial\hat{\Omega}$ with the probability $\mathcal{P}^{\hat{\Omega}}(\boldsymbol{p} \to \boldsymbol{s})$

```
SamplePoisson(p; Ω̂, ϵ)
begin
    q ← p;
    while q ∉ Ω̂_ϵ do
        Let B_q be the largest ball centered at q inside Ω̂;
        Draw q′ uniformly from the boundary ∂B_q of the ball B_q;
        q ← q′;
    end
    Compute the projection s of q on the boundary ∂Ω̂;
    return s;
end
```

When ϵ approaches zero, according to Kakutani's principle [Kakutani 1944], the outcome $\boldsymbol{s}$ of Algorithm 1 is distributed with a probability density that equals $\mathcal{P}^{\hat{\Omega}}(\boldsymbol{p} \to \boldsymbol{s})$. In practice, using $\epsilon > 0$ leads to a bias of order $O(1/\log \epsilon)$ [Binder and Braverman 2012], which is negligible when ϵ is small.

5.2 Estimating Gradient At Domain Boundary

The second step (**S.2**) of our method requires estimating the gradient $\nabla_{\hat{\Omega}} \hat{u}$, which equals ∇u at $\theta = 0$, at the point $\boldsymbol{s}$ sampled on the boundary (§5.1).

Let $\nabla_{\partial\hat{\Omega}} u$ and $\partial_{\boldsymbol{n}(\boldsymbol{s})} u$ denote the (vector-valued) *surface gradient* and (scalar-valued) *normal derivative* of u, respectively. Then, for all point $\boldsymbol{s}$ on the boundary $\partial\hat{\Omega} = \partial\Omega(0)$, it holds that

$$\nabla u(\boldsymbol{s}) = \nabla_{\partial\hat{\Omega}} u(\boldsymbol{s}) + \boldsymbol{n}(\boldsymbol{s})\, \partial_{\boldsymbol{n}(\boldsymbol{s})} u, \tag{23}$$

where $\boldsymbol{n}(\boldsymbol{s})$ denotes the inward unit-normal of $\partial\hat{\Omega}$ at $\boldsymbol{s}$.

On the right-hand side of Eq. (23), the surface gradient $\nabla_{\partial\hat{\Omega}} u(\boldsymbol{s})$ equals the gradient $\nabla g(\boldsymbol{s})$ of the boundary function g projected onto the tangent space at $\boldsymbol{s}$. Precisely,

$$\nabla_{\partial\hat{\Omega}} u(\boldsymbol{s}) = \nabla g(\boldsymbol{s}) - \boldsymbol{n}(\boldsymbol{s}) \cdot \nabla g(\boldsymbol{s}). \tag{24}$$

With the surface gradient $\nabla_{\partial\hat{\Omega}} u(\boldsymbol{s})$ obtained, our problem of computing the gradient $\nabla u(\boldsymbol{s})$ boils down to estimating the normal derivative $\partial_{\boldsymbol{n}(\boldsymbol{s})} u$. To this end, we introduce a new Monte Carlo estimator.

Let $B_{\boldsymbol{c}} \subset \hat{\Omega}$ be a ball that is tangent to the domain boundary $\partial\hat{\Omega}$ at $\boldsymbol{s}$ and centered at some fixed $\boldsymbol{c} \in \hat{\Omega}$ (see Figure 2), and $G^{B_{\boldsymbol{c}}}$ and $P^{B_{\boldsymbol{c}}}$ be the Green's function and Poisson kernel associated with $B_{\boldsymbol{c}}$. Previously, Sawhney et al. [2022] have shown that both $G^{B_{\boldsymbol{c}}}$ and $P^{B_{\boldsymbol{c}}}$ have analytical expressions. Please refer to their work for the exact forms of these kernels.

Our Monte Carlo estimator of the normal derivative $\partial_{\boldsymbol{n}(\boldsymbol{s})} u$ is based on the relation:

$$\partial_{\boldsymbol{n}(\boldsymbol{s})} u = \underbrace{\int_{B_{\boldsymbol{c}}} f(\boldsymbol{y})\, P^{B_{\boldsymbol{c}}}(\boldsymbol{y} \to \boldsymbol{s})\, \mathrm{d}\boldsymbol{y}}_{\text{interior}} + \underbrace{\int_{\partial B_{\boldsymbol{c}}} u(\boldsymbol{z})\, \partial_{\boldsymbol{n}(\boldsymbol{s})} P^{B_{\boldsymbol{c}}}(\boldsymbol{s} \to \boldsymbol{z})\, \mathrm{d}\boldsymbol{z}}_{\text{boundary}}. \tag{25}$$

We derive this relation and formally define the differential kernel $\partial_{\boldsymbol{n}(\boldsymbol{s})} P^{B_{\boldsymbol{c}}}(\boldsymbol{s} \to \boldsymbol{z})$ in §2.1 of the supplemental document.

Eq. (25) can be estimated using Monte Carlo integration. Unfortunately, a naïve implementation would suffer from high variance due to the singularities of the Poisson kernel $P^{B_{\boldsymbol{c}}}(\boldsymbol{y} \to \boldsymbol{s})$ at $\boldsymbol{y} = \boldsymbol{s}$ and the differential kernel $\partial_{\boldsymbol{n}(\boldsymbol{s})} P^{B_{\boldsymbol{c}}}(\boldsymbol{s} \to \boldsymbol{z})$ at $\boldsymbol{z} = \boldsymbol{s}$.

To address this problem, we introduce control variates to Eq. (25), yielding:

$$\partial_{\boldsymbol{n}(\boldsymbol{s})} u = \underbrace{\int_{B_{\boldsymbol{c}}} (f(\boldsymbol{y}) - f(\boldsymbol{s}))\, P^{B_{\boldsymbol{c}}}(\boldsymbol{y} \to \boldsymbol{s})\, \mathrm{d}\boldsymbol{y} + f(\boldsymbol{s})\, \frac{R}{n}}_{\text{interior}} + \underbrace{\int_{\partial B_{\boldsymbol{c}}} (u(\boldsymbol{z}) - g(\boldsymbol{s}))\, \partial_{\boldsymbol{n}(\boldsymbol{s})} P^{B_{\boldsymbol{c}}}(\boldsymbol{s} \to \boldsymbol{z})\, \mathrm{d}\boldsymbol{z}}_{\text{boundary}}, \tag{26}$$

where $R = \|\boldsymbol{c} - \boldsymbol{s}\|$ is the radius of the ball $B_{\boldsymbol{c}}$, and n denotes the dimensionality of the domain. Compared to Eq. (25), Eq. (26) has the advantage that the integrands of its *interior* and *boundary* components converge at $\boldsymbol{y} \to \boldsymbol{s}$ and $\boldsymbol{z} \to \boldsymbol{s}$, respectively. We show a derivation of this result (as well as a proof of the convergence) in §2.2 and §2.3 of the supplemental document.

We estimate Eq. (26) using Monte Carlo integration, as summarized in Algorithm 2. In practice, we draw $\boldsymbol{y}$ and $\boldsymbol{z}$ uniformly from the ball $B_{\boldsymbol{c}}$ and its surface $\partial B_{\boldsymbol{c}}$, respectively. Further, we utilize antithetic sampling for the *boundary* component by accompanying each sampled $\boldsymbol{z}$ an antithesis $\boldsymbol{z}^*$ obtained by mirroring $\boldsymbol{z}$ around the line connecting $\boldsymbol{s}$ and $\boldsymbol{c}$. Precisely,

$$\boldsymbol{z}^* = \boldsymbol{s} + 2\,(\boldsymbol{n}(\boldsymbol{s}) \cdot \Delta\boldsymbol{z})\, \boldsymbol{n}(\boldsymbol{s}) - \Delta\boldsymbol{z}, \tag{27}$$

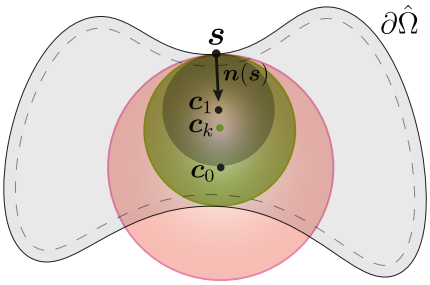

Figure 3: We search for the largest ball—which is $B_{\boldsymbol{c}_k}$ in this example—inside the domain $\hat{\Omega}$ with the center located along the ray $(\boldsymbol{s}, \boldsymbol{n}(\boldsymbol{s}))$. To this end, we use bisection to find the maximal radius t of the ball based on Eq. (28).

ALGORITHM 2: Estimating the normal derivative $\partial_{\boldsymbol{n}(\boldsymbol{s})} u$ at a boundary point $\boldsymbol{s}$

```
EstNormalDerivative(s; Ω̂, ε₀)
begin
    Find c ∈ Ω̂ such that the ball B_c is tangent to ∂Ω̂ at s;
    ∂_{n(s)} u = 0;
    /* Estimating the interior component                    */
    Draw y from B_c with the probability p(y);
    ∂_{n(s)} u += [(f(y) − f(s)) P^{B_c}(y → s)]/p(y) + f(s) R/n;
    /* Estimating the boundary component                    */
    Draw z from ∂B_c with the probability p(z);
    if ‖z − s‖ > ε₀ then
        Compute the antithetic sample z*;
        Estimate u_z = u(z), u_{z*} = u(z*) using WoS ; // Eq. (6)
        ∂_{n(s)} u += [((u_z + u_{z*})/2 − g(s)) ∂_{n(s)} P^{B_c}(s → z)]/p(z);
    end
    return ∂_{n(s)} u;
end
```

where $\Delta\boldsymbol{z} \coloneqq \boldsymbol{z} - \boldsymbol{s}$. To avoid numerical issues, we discard samples of $\boldsymbol{z}$ that are too close to $\boldsymbol{s}$ (i.e., when $\|\boldsymbol{z} - \boldsymbol{s}\| < \epsilon_0$ for some small ϵ_0).

Lastly, as we will demonstrate in §6, making the ball $B_{\boldsymbol{c}}$ (that is tangent to the boundary $\partial\hat{\Omega}$ at $\boldsymbol{s}$) larger usually leads to lower variance. To this end, we seek a large $t > 0$ such that the ball centered at $\boldsymbol{c} = \boldsymbol{s} + t\,\boldsymbol{n}(\boldsymbol{s})$ with radius t is fully contained in the domain $\hat{\Omega}$ and its boundary $\partial\hat{\Omega}$. It is easy to verify that this holds if and only if

$$r^{\hat{\Omega}}(\boldsymbol{s} + t\,\boldsymbol{n}(\boldsymbol{s})) = t, \tag{28}$$

with $r^{\hat{\Omega}}(\boldsymbol{p})$ denoting the minimal distance from $\boldsymbol{p}$ to the boundary $\partial\hat{\Omega}$ for any $\boldsymbol{p} \in \hat{\Omega}$. On the other hand, if the ball goes beyond $\hat{\Omega} \cup \partial\hat{\Omega}$, one would have $r^{\hat{\Omega}}(\boldsymbol{s} + t\,\boldsymbol{n}(\boldsymbol{s})) < t$ instead.

In practice, based on the relation in Eq. (28), we search for the greatest t using bisection (see Figure 3).

Discussion. Previous methods [Sawhney and Crane 2020; Miller et al. 2023] estimate normal derivatives at boundary points based

on the following relation:

$$\nabla u(\boldsymbol{x}) = \underbrace{\int_{B_{\boldsymbol{x}}} f(\boldsymbol{y})\, \frac{\partial}{\partial \boldsymbol{x}} G(\boldsymbol{x} \leftrightarrow \boldsymbol{y})\, \mathrm{d}\boldsymbol{y}}_{\text{interior}} + \underbrace{\frac{1}{|B_{\boldsymbol{x}}|} \int_{\partial B_{\boldsymbol{x}}} u(\boldsymbol{z})\, \frac{\boldsymbol{z}-\boldsymbol{x}}{R}\, \mathrm{d}\boldsymbol{z}}_{\text{boundary}}, \tag{29}$$

where R denotes the radius of the ball $B_{\boldsymbol{x}}$. Specifically, the normal derivative $\partial_{\boldsymbol{n}(\boldsymbol{s})} u$ is computed approximately via

$$\partial_{\boldsymbol{n}(\boldsymbol{s})} u = \boldsymbol{n}(\boldsymbol{s}) \cdot \nabla u(\boldsymbol{s}) \approx \boldsymbol{n}(\boldsymbol{s}) \cdot \nabla u(\boldsymbol{s}_l), \tag{30}$$

where $\boldsymbol{s}_l := \boldsymbol{s} + l\,\boldsymbol{n}(\boldsymbol{s})$ is a point in the interior of the domain Ω (for some small $l > 0$), and $\nabla u(\boldsymbol{s}_l)$ is estimated by applying Monte Carlo integration to Eq. (29) (by letting $\boldsymbol{x} = \boldsymbol{s}_l$ and having $u(\boldsymbol{z})$ on the right-hand side computed using walk-on-spheres).

We will demonstrate in §6 that, compared with previous methods, our estimator described in Eq. (26) and Algorithm 2 can produce significantly lower variance.

6 RESULTS

We implement our estimator expressed in §5 on the GPU with Dr.Jit [Jakob et al. 2022] as the numerical backend.

6.1 Normal-Derivative Estimator

We first evaluate the effectiveness of our normal-derivative estimator outlined in Algorithm 2.

Baseline method. For this evaluation, we use the approach given by Eqs. (29) and (30) as the baseline. As suggested by Sawhney and Crane [2020], this baseline method leverages moving control variates and antithetic sampling. We note that, although our normal-derivative estimator also uses control variates and antithetic sampling, they are applied very differently.

Ablation. We present an ablation study in Figure 4 that compares the performance of four normal-derivative estimators including the baseline method described above (indicated as "baseline"), our full method (indicated as "ours"), our method but without performing antithetic sampling of $\boldsymbol{z}$ (indicated as "ours (no anti.)"), and our method using a smaller ball $B_{\boldsymbol{c}}$ whose radius equals 20% of the maximum (indicated as "ours (small ball)").

In this ablation, we apply all four methods to a 2D Laplace problem over a disc (1a–1c) and a 2D Poisson problem over a clover shape (2a–2c). For both problems, we estimate normal derivatives at a single evaluation point using high sample counts (i.e., 2000 and 100000, respectively). We show the cumulative average of sample contributions in (b), and the variance of all samples in (c).

For the first problem, our full method enjoys near-zero variance by using the ball $B_{\boldsymbol{c}}$ that equals the entire domain $\hat{\Omega}$. Even by using a small ball, our method outperforms the baseline method by having less than a third of the variance.

The second problem has a concave domain, yielding a smaller ball $B_{\boldsymbol{c}}$ at the evaluation point. Still, all variants of our method outperform the baseline with the full version being the clear winner.

6.2 Full Estimator

We now evaluate our full WoS estimator presented in §5.

Baseline method. As discussed in §4, a naïve way to formulate the derivative ∂u is to differentiate the integral equation (5) using Reynolds transport theorem [1903]. When the domain Ω depends on the parameter θ, so will the ball $B_{\boldsymbol{x}}$ (and its boundary $\partial B_{\boldsymbol{x}}$), causing the derivative of the *interior* component of Eq. (5) to involve a *boundary* integral capturing the rate at which the ball grows or shrinks with respect to θ.

To demonstrate the advantage of our formulation introduced in §4, we implement an additional estimator using this naïve formulation as a baseline.

Validation and evaluation. In Figure 5, we evaluate our technique by comparing derivatives (with respect to translations of the boundaries) estimated by our estimator to those obtained with finite differences (FD) and the baseline method. The first two examples ("Wrench" and "Teapot") use 2D Laplace problems (i.e., with zero source), while the third example ("Globe") uses a 2D Poisson problem with an analytical source formulated by a 2D sinusoidal wave. The last example ("Bunny") uses a 3D Laplace problem. For all four examples, our results closely match the FD references and are significantly cleaner than the results generated with the baseline method in equal time.

Solving inverse problems. Lastly, in Figures 6–8, we solve inverse Poisson problems using gradients generated with our method and the baseline. Specifically, we infer parameters related to the shape of the domain Ω by minimizing the difference between the target and the simulated solution u (observed at all or part of the domain). We use the Adam algorithm [Kingma and Ba 2014] with identical configurations (e.g., initialization and learning rate) for the optimizations.

Figure 6 includes two examples ("Wrench" and "Globe") that use, respectively, inverse 2D Laplace and Poisson problems. For each example, we search for the rotation angle of the domain boundary based on the solution u (sampled as an image).

Figure 7 contains a "Bunny" example that uses an inverse (exterior) Laplace problem in 3D with a bunny-shaped domain. We optimize the pose of the bunny based on solution values sampled over the surface of a cube surrounding the domain.

Figure 8 shows a "Diffusion curve" example that involves an inverse Laplace problem in 2D where the shape of the domain boundary (expressed using a polyline) is freely optimized based on the solution u sampled as an image.

In all cases, our estimator allows the optimizations to converge to the ground truth smoothly, while the baseline method fails due to high variance.

7 DISCUSSION AND CONCLUSION

Limitations and future work. When performing forward solve, our implementation uses the "vanilla" walk-on-spheres (WoS) method by Sawhney and Crane [2020]. Incorporating more advanced variants (e.g., [Qi et al. 2022; Miller et al. 2023]) or walk-on-boundary (WoB) methods [Sugimoto et al. 2023] into our pipeline may further improve its efficiency.

Also, when estimating normal derivatives using Eq. (26), we sample both the *interior* and the *boundary* integrals uniformly.

Devising better sampling techniques for these integrals can be beneficial.

Lastly, our technique focuses on Poisson equations with Dirichlet boundary conditions. Generalizing our method to support a wider range of PDEs (e.g., screened Poisson) and other (e.g., Neumann) boundary conditions is an important topic for future research.

Conclusion. We introduced a theory that differentiates solutions to the Poisson equation with Dirichlet boundary conditions. Specifically, we devised a new formulation that expresses the derivatives as *boundary* integrals and offers the generality of differentiating with respect to arbitrary parameters including domain shapes.

Based on this formulation, we developed a Monte Carlo estimator based on the grid-free walk-on-spheres (WoS) process. At the core of our technique is a new method that estimates normal derivatives of the solution (at domain boundaries) by leveraging a new form of control variates and antithetic sampling.

Compared with prior methods, our WoS estimator is capable of providing derivative estimates with significantly lower variances—which we demonstrated via several differentiable and inverse PDE problems.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive suggestions. We are also grateful to Aaron Lefohn for his support. This work started when Zihan Yu was an intern at NVIDIA. Zihan Yu and Zhiqian Zhou's contributions while at the University of California, Irvine were partially supported by NSF grant 1900927.

REFERENCES

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

J.M Delaurentis and L.A Romero. 1990. A Monte Carlo method for Poisson's equation. *J. Comput. Phys.* 90, 1 (1990), 123–140.

Michel Delfour and Jean-Paul Zolsio. 2010. *Shapes and Geometries: Metrics, Analysis, Differential Calculus, and Optimization.* Society for Industrial and Applied Mathematics.

L.C. Evans. 2010. *Partial Differential Equations.* American Mathematical Society.

Jaroslav Haslinger and Raino AE Mäkinen. 2003. *Introduction to shape optimization: theory, approximation, and computation.* SIAM.

Dan Henry. 2005. *Perturbation of the boundary in boundary-value problems of partial differential equations.* Number 318. Cambridge University Press.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022. DR.JIT: a just-in-time compiler for differentiable rendering. *ACM Trans. Graph.* 41, 4 (2022), 124:1–124:19.

Shizuo Kakutani. 1944. 143. Two-dimensional Brownian Motion and Harmonic Functions. *Proceedings of the Imperial Academy* 20, 10 (1944), 706–714.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2023. Boundary Value Caching for Walk on Spheres. *ACM Trans. Graph.* 42, 4 (2023), 82:1–82:11.

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

Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph.* 27, 3 (2008), 1–8.

Yang Qi, Dario Seyb, Benedikt Bitterli, and Wojciech Jarosz. 2022. A bidirectional formulation for Walk on Spheres. *Computer Graphics Forum* 41, 4 (2022), 51–62.

O. Reynolds. 1903. *Papers on mechanical and physical subjects: the sub-mechanics of the universe.* Vol. 3. The University Press.

Karl Karlovich Sabelfeld. 1982. Vector algorithms in the Monte-Carlo method for solving systems of second-order elliptic equations and Lame's equation. In *Doklady Akademii Nauk*, Vol. 262. Russian Academy of Sciences, 1076–1080.

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), 123:1–123:18.

Rohan Sawhney, Bailey Miller, Ioannis Gkioulekas, and Keenan Crane. 2023. Walk on Stars: A Grid-Free Monte Carlo Method for PDEs with Neumann Boundary Conditions. *ACM Trans. Graph.* 42, 4 (2023), 80:1–80:20.

Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. 2022. Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients. *ACM Trans. Graph.* 41, 4 (2022), 53:1–53:17.

Jan Sokolowski and Jean-Paul Zolésio. 1992. *Introduction to shape optimization.* Springer.

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 (2023), 81:1–81:16.

Shawn Walker. 2015. *The Shapes of Things: A Practical Guide to Differential Geometry and the Shape Derivative.* Society for Industrial and Applied Mathematics.

Ekrem Fatih Yilmazer, Delio Vicini, and Wenzel Jakob. 2022. Solving inverse PDE problems using grid-free Monte Carlo estimators. https://arxiv.org/abs/2208.02114

Shuang Zhao, Frédo Durand, and Changxi Zheng. 2018. Inverse Diffusion Curves Using Shape Optimization. *IEEE Transactions on Visualization and Computer Graphics* 24, 7 (2018), 2153–2166.

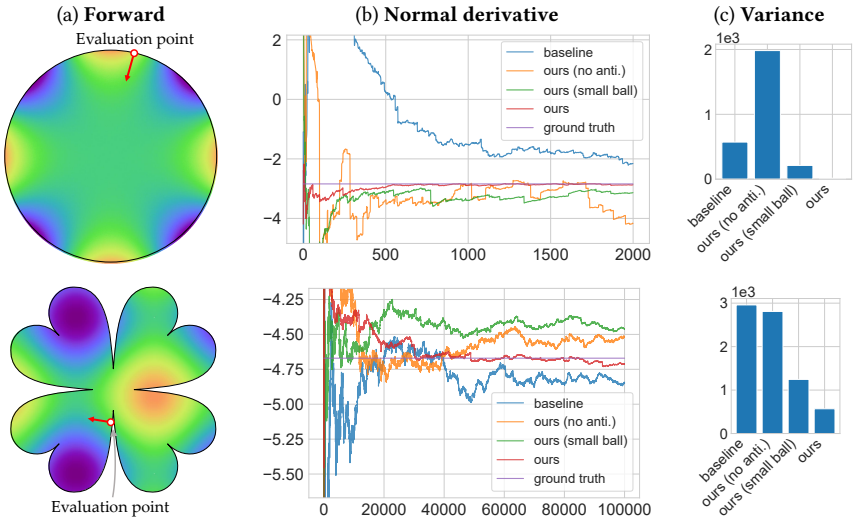

Figure 4: Ablation: We compare the performance of four normal-derivative estimators: the baseline method by Sawhney and Crane [2020], ours without antithetic sampling, ours using a small ball B_c, and our full method. We apply all four estimators to a Laplace problem over a disc (top) and a Poisson problem over a clover (bottom). For both problems, we estimate normal derivative at one evaluation point.

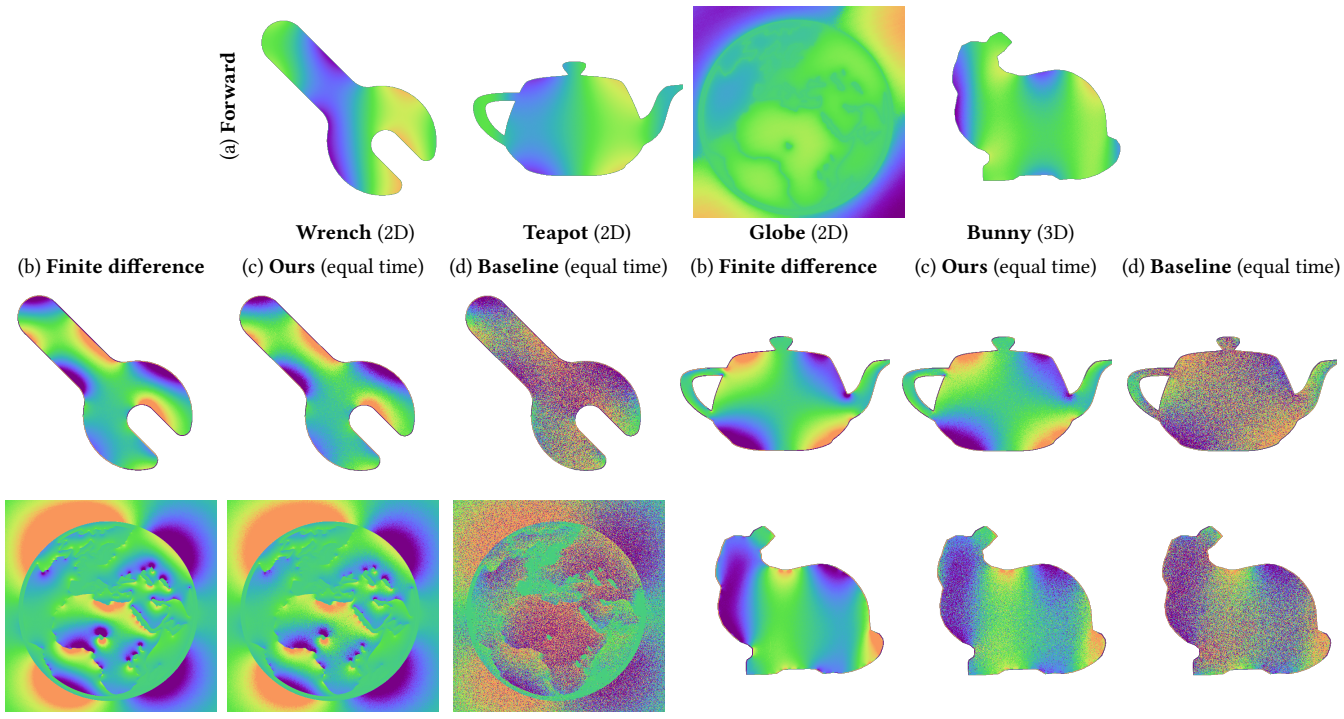

Figure 5: Differentiable PDE solve results: (a) Solutions to Poisson equations with Dirichlet boundary conditions (i.e., forward solve results); (b) Derivatives computed using finite differences (FD); (c, d) Derivatives estimated in equal time using our technique and the baseline estimator (described in §6.2). All images in this figure use the same color map as Figure 1.

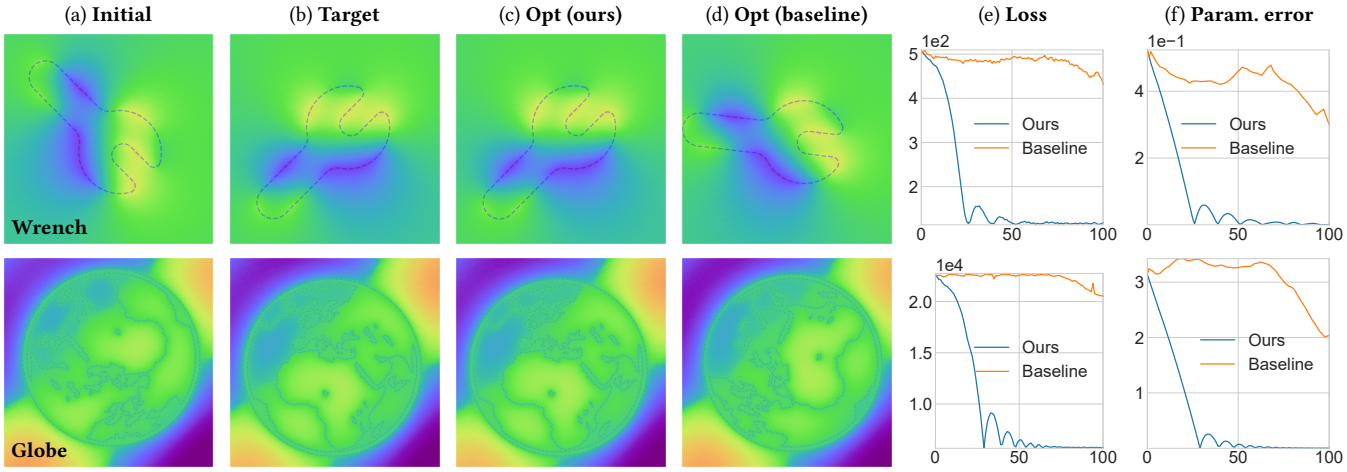

Figure 6: Solving 2D inverse problems: For both examples, we optimize the rotation angle of the domain boundary based on the solution field u near the boundary (both interior and exterior). All images in this example use the same color map as Figure 1. The parameter error in (f) is used only for evaluation (and not for optimization).

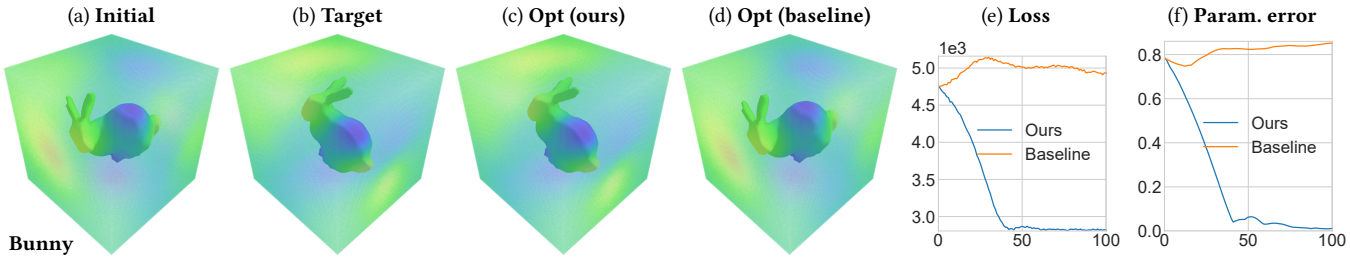

Figure 7: Solving 3D inverse problem: In this example, we infer the pose of a 3D bunny (with fixed Dirichlet boundary conditions over the surface) based on the solution u sampled on the surface of a cube surrounding the bunny. All images in the top row use the same color map as Figure 1. The parameter error in (f) is used only for evaluation (and not for optimization).

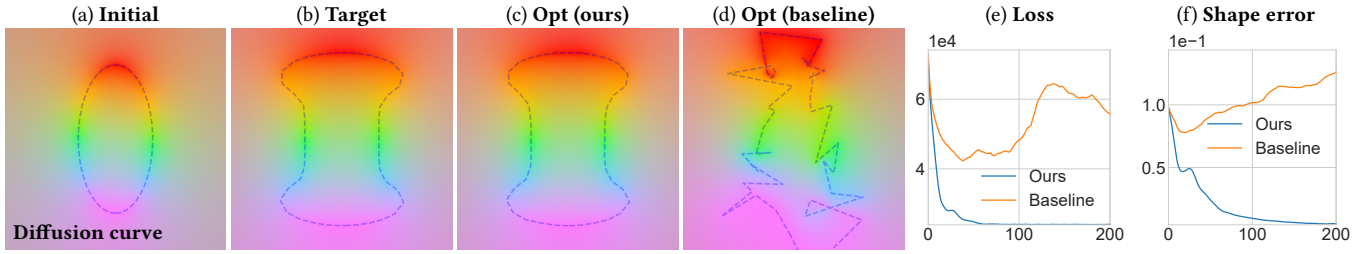

Figure 8: Solving 2D shape-optimization problem: In this example, we optimize the shape of the domain boundary (expressed as per-vertex positions) based on the colored (i.e., RGB) solution field u around the boundary. The shape error in (f) is used only for evaluation (and not for optimization).