Solving Inverse PDE Problems using Grid-Free Monte Carlo Estimators

EKREM FATIH YILMAZER, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
DELIO VICINI, Google Inc., Switzerland
WENZEL JAKOB, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

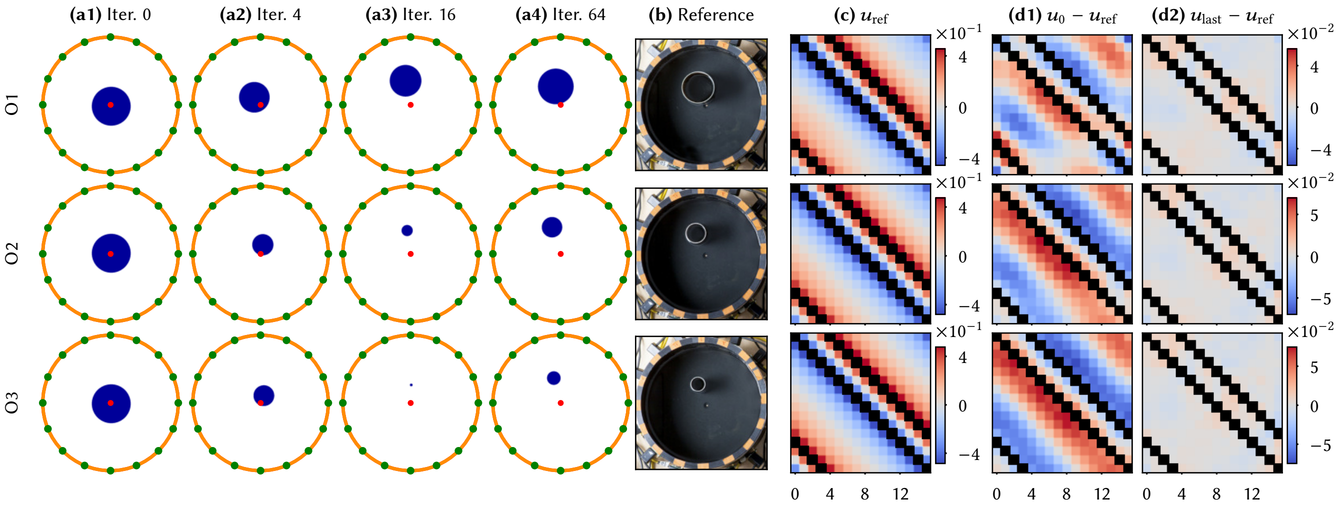


Fig. 1. We apply our inverse PDE solver to a 2D *electrical impedance tomography* experiment [Hauptmann et al. 2017], in which an electric current flows through a saline-filled water tank containing conducting objects of different sizes (photographs in middle). The marker in the middle indicates the center. Injecting a current using 2 of 16 uniformly spaced electrodes at the tank boundary generates a measurable voltage difference at the other electrodes, and injecting at various locations produces a matrix u_{ref} of measurements. The objective of this inverse problem is to infer the properties of the conductor from this data. We perform a differentiable simulation of this setup to optimize the center and radius of a conducting circle. The frames on the left show the progression of the optimization (columns **(a)**), while the rightmost two columns reveal how the predicted voltages become increasingly consistent with the measurement (columns **(d)**).

Partial differential equations can model diverse physical phenomena including heat diffusion, incompressible flows, and electrostatic potentials. Given a description of an object's boundary and interior, traditional methods solve such PDEs by densely meshing the interior and then solving a large and sparse linear system derived from this mesh. Recent grid-free solvers take an alternative approach and avoid this complexity in exchange for randomness: they compute stochastic solution estimates and generally bear a striking resemblance to physically-based rendering algorithms.

In this article, we develop algorithms targeting the inverse form of this problem: given an already existing solution of a PDE, we infer parameters characterizing the boundary and interior. In the grid-free setting, there are again significant connections to rendering, and we show how insights from both fields can be combined to compute unbiased derivative estimates that enable gradient-based optimization. In this process, we encounter new challenges that must be addressed to obtain practical solutions. We introduce acceleration and variance reduction strategies and show how to differentiate branching random walks in reverse mode.

We finally demonstrate our approach on both simulated data and a real-world electrical impedance tomography experiment, where we reconstruct the position of a conducting object from voltage measurements taken in a saline-filled tank.

Authors' Contact Information: Ekrem Fatih Yilmazer, ekrem.yilmazer@epfl.ch, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland; Delio Vicini, vicini@google.com, Google Inc., Zurich, Switzerland; Wenzel Jakob, wenzel.jakob@epfl.ch, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.

CCS Concepts: • **Mathematics of computing → Partial differential equations**; • **Computing methodologies → Rendering**.

Additional Key Words and Phrases: Walk on Spheres, Differentiable Rendering, Path Replay Backpropagation, Electrical Impedance Tomography

ACM Reference Format:
Ekrem Fatih Yilmazer, Delio Vicini, and Wenzel Jakob. 2024. Solving Inverse PDE Problems using Grid-Free Monte Carlo Estimators. *ACM Trans. Graph.* 43, 6, Article 175 (December 2024), 18 pages. https://doi.org/10.1145/3687990

1 INTRODUCTION

Many physical phenomena are naturally described using partial differential equations (PDEs). For example, the heat equation models the spread of thermal energy in a potentially heterogeneous material. Solvers that numerically approximate solutions of such PDEs are in widespread use. We pursue the opposite direction in this article, which is known as an *inverse PDE problem*: estimating unknown parameters from observations of the solution. This set of unknown parameters could include various PDE coefficients, boundary conditions, and even the shape of the domain.

Such problems arise in diverse scientific and engineering contexts, for example to determine the physical parameters of a thermal conductor from measurements [Cannon 1964]. Electrical impedance tomography [Cheney et al. 1999] seeks to reconstruct the interior of a living organism. Electrodes provide measurements of the electric field, which is influenced by the tissue's conductivity, impedance, and dielectric permittivity.

Our approach entails differentiating the solver and recovering the unknown parameters using gradient descent. However, one issue with conventional PDE solvers based on the finite element method (FEM) is that they require a meshing step that can be fragile and computationally costly. An alternative is using Monte Carlo PDE solvers based on the *walk on spheres* (WoS) [Muller 1956]. These *grid-free* methods sample random paths in the domain to compute unbiased estimates of the solution. Grid-free solvers have recently attracted significant attention in the computer graphics community, partly owing to the similarities to rendering [Sawhney and Crane 2020] and the algorithmic synergies that this creates [Sawhney et al. 2022, 2023; Qi et al. 2022; Bati et al. 2023].

A common issue with gradient-based optimization is that the standard approach for reverse-mode differentiation (also known as *backpropagation*) reverses the data dependencies of an underlying computation. When applied to the WoS algorithm, this means that intermediate results of a large number of iterations would need to be stored to enable the subsequent differentiation.

In rendering, recent progress has led to the development of *differentiable rendering* methods [Gkioulekas et al. 2013; Li et al. 2018; Nimier-David et al. 2019] that estimate derivatives of complete light transport simulations. A similar issue arises here as well: light paths can potentially be very long, particularly in highly-scattering media, which makes naïve differentiation prohibitively memory-intensive. This has sparked work on specialized *adjoint methods* [Nimier-David et al. 2020; Vicini et al. 2021] that cast differentiation into an independent simulation of "derivative light" to avoid these costs.

We demonstrate that this idea also applies to Monte Carlo PDE solvers, where it leads to unbiased derivative estimators for several variants and extensions of the WoS algorithm. At the same time, unique characteristics of this setting require the development of new solutions: for example, PDEs with spatially variable coefficients often require a variance reduction strategy where random walks can split. We extend *Path Replay Backpropagation* [Vicini et al. 2021] to branching random walks to support this requirement.

In our experiments, we found that the forward simulation was often the main computational bottleneck, and we propose several improvements to accelerate convergence of both primal and adjoint simulations: first, to better handle PDEs with a heterogeneous diffusion coefficient, we introduce an adaptive sphere radius selection. Second, we develop a fast way to evaluate and sample the Green's function, accelerating convergence by up to an order of magnitude.

As an application of our work, we investigate electrical impedance tomography (EIT) [Cheney et al. 1999]. Our method provides great flexibility in the parameterization of the unknown conductance sought by EIT that we demonstrate in recovering continuously varying and binary conductance fields. Finally, we use our solver to reconstruct the shape and position of a conducting object in a saline-filled tank (Figure 1). This demonstrates that our approach can translate to real-world data despite modeling approximations.

In summary, our contributions are:

- We develop efficient adjoints for the Walk on Spheres [Sawhney and Crane 2020] and Walk on Stars [Sawhney et al. 2023], demonstrating recovery of governing parameters (e.g., diffusion coefficient, source terms).
- We then extend these methods to branching random walks.
- We improve the performance of the forward PDE solvers by introducing an adaptive sphere radius selection and optimized Green's function evaluation.
- We apply these ideas to electrical impedance tomography and present results on measured data.

The experiments in this paper use 2D domains for ease of visualization, though the presented ideas straightforwardly generalize to higher dimensions. We will release an open source implementation of our method to ensure reproducibility.

2 RELATED WORK

We briefly review prior work on Monte Carlo PDE solvers, differentiable Monte Carlo estimators and inverse PDE problems.

Monte Carlo PDE solvers. The solvers in this paper all build on variations of *walk on spheres* [Muller 1956]. Sawhney and Crane [2020] were the first to explore this technique in context of computer graphics, realizing their relationship to existing Monte Carlo rendering methods [Pharr et al. 2023]. Unique characteristics of these solvers are that they provide statistically unbiased solution estimates with a computation that is local and progressive in nature, all without requiring a mesh of the domain's interior. This makes it possible to solve and even interactively preview problems on large and complex domains. The base algorithm was later extended to spatially-varying coefficients [Sawhney et al. 2022], as well as Neumann [Sawhney et al. 2023] and Robin boundary conditions [Miller et al. 2024]. Other recent uses of Monte Carlo PDE solvers include estimators for fluid motion [Rioux-Lavoie et al. 2022] and coupled solvers for thermal transport [Bati et al. 2023].

As in Monte Carlo rendering, the stochastic nature of estimates motivates specialized variance reduction strategies. Qi et al. [2022] propose a bidirectional estimator for PDEs with sparse source terms. Other works cache and reuse evaluations on the boundary [Miller et al. 2023] or interior [Bakbouk 2023] or approximate the solution using a neural cache [Li et al. 2023]. Nabizadeh et al. [2021] warp space inside-out to efficiently solve PDEs on unbounded domains.

Differentiable Monte Carlo methods. Given the origins of the Monte Carlo method in the area of nuclear physics, it is of no surprise that the first uses of *differential Monte Carlo* likewise investigated nuclear processes, for example to determine the efficiency of neutron breeding [Mikhailov 1967], design radiation shielding [Brainina et al. 1967], or model the criticality transition in nuclear reactors [Lux and Koblinger 1990]. In computer graphics, the use of differential Monte Carlo was initially focused on appearance estimation [Gkioulekas et al. 2013, 2016; Khungurn et al. 2015; Velinov et al. 2018]. Recently, general algorithms [Li et al. 2018] and systems [Nimier-David et al. 2019; Jakob et al. 2022] have enabled more systematic use of differentiable Monte Carlo simulations. In rendering, special consideration is required to differentiate visibility discontinuities [Li et al. 2018; Loubet et al. 2019; Zhang et al. 2020; Vicini et al. 2022]. Concurrently to our work, Yu et al. [2024] discuss application of some of those ideas to optimize the boundary shape of Poisson problems. The non-trivial control flow and high number of operations found in rendering algorithms make conventional automatic differentiation impractical, which has motivated the development of more efficient adjoint methods [Nimier-David et al. 2020; Vicini et al. 2021] that we also build upon.

Inverse PDE problems. PDE-constrained inverse problems occur in a wide range of practical applications such as electrical impedance tomography [Barber and Brown 1984; Adler and Boyle 2017], fluid control [McNamara et al. 2004], geophysics [Mosser et al. 2020], nanophotonics [Molesky et al. 2018] and thermal design [França et al. 2003]. In computer graphics, Zhao et al. [2018] solve an inverse problem to control diffusion curves [Orzan et al. 2013].

Electrical impedance tomography. As an example application, we demonstrate the use of our solvers for electrical impedance tomography. Prior work can be categorized into statistical and deterministic approaches. Statistical methods are mainly based on Bayesian inversion [Liu et al. 2018, 2020a,b]. Deterministic approaches either assume linearity [Santosa and Vogelius 1990; Barber and Brown 1984; Calderón 2006; Cheney et al. 1990; Adler et al. 2009], or use iterative algorithms such as Gauss-Newton or Newton-Raphson [Bayford 2006]. An example of a direct non-linear method is the D-Bar algorithm [Hamilton et al. 2012, 2018]. Other approaches use level set methods [Liu et al. 2017], morphable components [Liu and Du 2021] or directly predict conductivities using neural networks [Guardo et al. 1991; Li et al. 2017; Zhang et al. 2022].

3 BACKGROUND: MONTE CARLO PDE SOLVERS

The discussion of our method (Section 4) assumes familiarity with Monte Carlo PDE solvers that we review here for completeness. Table 1 lists notation used throughout the paper. For a more thorough introduction, we also refer to the original publications [Sawhney and Crane 2020; Sawhney et al. 2022, 2023].

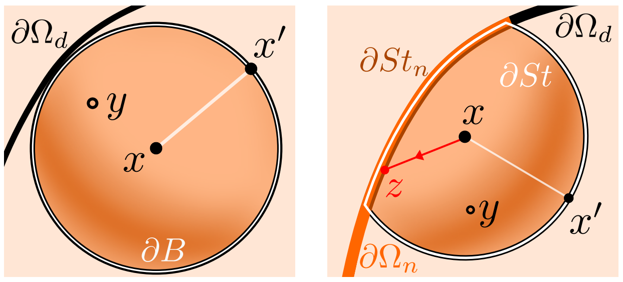
3.1 Walk on Spheres

Screened Poisson equation. We consider the screened Poisson equation with Dirichlet boundary conditions as an introductory problem:

$$\begin{aligned} \Delta u(x) - \sigma u(x) &= -f(x) \quad && x \in \Omega, \\ u(x) &= g(x) \quad && x \in \partial\Omega. \end{aligned} \tag{1}$$

Here, $u(x)$ denotes the solution at a point inside the domain Ω. The function $f(x)$ is a spatially-varying *source term*, and $g(x)$ prescribes the solution values on the domain boundary $\partial\Omega$. The scalar *screening coefficient* σ models a decay in the influence of distant sources and boundaries. Solutions to Equation (1) satisfy a generalized *mean value property* [Muller 1956; Sawhney and Crane 2020] that expresses the solution $u(x)$ in the form of an integral equation

$$u(x) = \int_{\partial B(x)} P^\sigma(x, x')\, u(x')\, \mathrm{d}x' + \int_{B(x)} G^\sigma(x, y)\, f(y)\, \mathrm{d}y, \tag{2}$$



(a) Sphere step (Dirichlet) (b) Star step (Dirichlet & Neumann)

Fig. 2. At every iteration we either take a sphere (a) or a star step (b) depending on the local boundary conditions. The former samples a position $x' \in \partial B$, where B is a ball centered at x and tangential to the domain's Dirichlet boundaries $\partial\Omega_d$. Another sample $y \in B$ may be needed in the presence of interior sources. Star steps (b) are required to handle Neumann boundary conditions. They construct the largest ball that is tangential to $\partial\Omega_d$ and involve integrals over the *star* $St = B \cap \Omega$, its boundary ∂St and the subset $\partial St_n = \partial St \cap \partial\Omega_n$ overlapping Neumann boundary conditions.

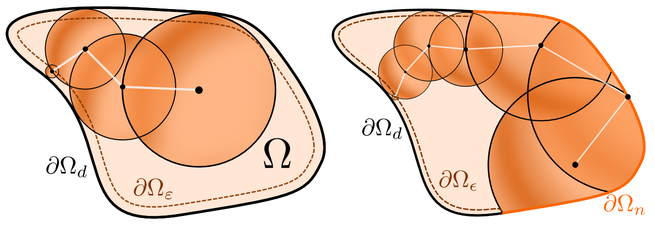
Table 1. Notation used throughout paper

Term	Interpretation
$\Omega, \partial\Omega$	The domain and its boundary
$B, \partial B$	The current ball and its boundary
$\lvert\partial B\rvert$	The surface area of the ball
$St, \partial St$	The current star and its boundary
$\partial\Omega_d$	Part of $\partial\Omega$ with Dirichlet boundary conditions
$\partial\Omega_n$	Part of $\partial\Omega$ with Neumann boundary conditions
∂St_n	Part of ∂St with Neumann boundary conditions
$f(y)$	Spatially varying source term at $x \in \Omega$
$g(x)$	Value of the solution at $x \in \partial\Omega_d$
$h(x)$	Normal derivative of the solution at $x \in \partial\Omega_n$
$\sigma(y)$	Spatially varying screening coefficient
$\alpha(y)$	Spatially varying diffusion coefficient
x, x'	Current and next vertex of the Walk on Spheres
P^σ, G^σ	Poisson kernel and Green's function

where $B(x) \subset \Omega$ is a ball around x, $P^\sigma(x, x')$ is the Poisson kernel and $G^\sigma(x, y)$ the Green's function. The Poisson kernel and Green's function depend on both σ and the radius of the ball—expressions for them can be found in the supplementary material.

Monte Carlo estimator. The integral form makes it possible to construct Monte Carlo estimators of $u(x)$. Starting at any point $x \in \Omega$, the *Walk on Spheres* [Muller 1956; Sawhney and Crane 2020] recursively estimates the solution by uniformly sampling a point $x' \in \partial B(x)$ (see Figure 2a). The source term's contribution is evaluated by sampling an additional point $y \in B(x)$ at every step. The position of y can be chosen proportional to the Green's function to reduce variance, and Section 5.2 introduces an efficient way to do so.

The radius at each step is arbitrary as long as the resulting ball does not intersect the boundary. It is usually set to the minimum distance to the boundary, as taking the largest admissible sphere



(a) Walk on Spheres (b) Walk on Stars

Fig. 3. (a) Each step of a Walk on Spheres samples a random point on the largest sphere that fits within the domain. The algorithm terminates once it reaches a position in Ω_ε. (b) In the Walk on Stars, the recursive integration over the solution on ∂St_n implies a "reflection" of the sampled path when encountering a Neumann boundary.

step improves efficiency. The algorithm terminates once it samples a point within a small distance ε to the boundary (Figure 3a).

3.2 Walk on Stars

Mixed boundary conditions. A generalization of the earlier family of PDEs additionally considers Neumann boundary conditions:

$$\begin{aligned} \Delta u(x) - \sigma u(x) &= -f(x) \quad && x \in \Omega, \\ u(x) &= \ g(x) && x \in \partial\Omega_d, \\ \partial_n u(x) &= \ h(x) && x \in \partial\Omega_n, \end{aligned} \tag{3}$$

where the sets $\partial\Omega_d$ and $\partial\Omega_n$ partition $\partial\Omega$ into regions with Dirichlet and Neumann boundary conditions, and $h(x)$ specifies the normal derivative $\partial_n u(x)$ on the boundary. We assume that the Neumann boundary is (locally) convex and smooth. Computing solutions on non-convex Neumann domains is more involved [Sawhney et al. 2023] and beyond the scope of this work.

As before, the problem can be cast into an equivalent recursive integral form. Instead of integrating over a ball, we must now integrate over a *star* defined as the intersection of the domain and a ball, whose radius equals the minimum distance between x and $\partial\Omega_d$ (see Figure 2b). This leads to the following *boundary integral equation* (BIE):

$$\begin{aligned} \kappa(x)u(x) &= \int_{\partial St} P^\sigma(x,x')u(x')\,\mathrm{d}x' - \int_{\partial St_n} G^\sigma(x,z)h(z)\,\mathrm{d}y \\ &+ \int_{St} G^\sigma(x,y)f(y)\,\mathrm{d}y \\ \kappa(x) &= 1/2 \quad x \in \partial\Omega_n, \quad \kappa(x) = 1 \quad x \notin \partial\Omega_n, \end{aligned} \tag{4}$$

where St denotes the star, ∂St is its boundary, and $\partial St_n = \partial St \cap \partial\Omega_n$ refers to the intersection of the star's boundary and the Neumann boundary region. If the closest point on the boundary is part of $\partial\Omega_d$, the expression reduces to Equation (2).

Monte Carlo estimator. Generalizing the previous estimator to Equation 4 leads to the *Walk on Stars* [Sawhney et al. 2023]. In each iteration, this method samples a random ray starting at x, whose direction is either uniform on the sphere (if $x \in \Omega$) or uniform on the hemisphere pointing towards the interior (if $x \in \partial\Omega$). Intersecting this ray with ∂St produces a sample x' that can be used to estimate the recursive term (first integral). To account for sources (third integral), the method picks a position along the ray proportional to the Green's function, ignoring samples that fall outside of the star.

To handle Neumann boundaries (second integral), Sawhney et al. [2023] construct a custom bounding volume hierarchy (BVH) to sample the domain ∂St_n. We use simpler scheme that we found to be more suitable for the Bézier spline curve representation used in this article. This method does not require custom data structures in exchange for a potentially larger amount of variance; we explain it in the supplemental material.

Figure 3b illustrates the complete estimator. At each step, we check the type of the closest boundary position to select between the sphere (Dirichlet) and star (Neumann) cases. We then take a suitable step, accumulating contributions according to Equations (2) and (4). This process repeats until reaching a position $\bar{x}_k$ within distance ε to a Dirichlet boundary, where the solution is approximated by $g(\bar{x}_k)$.

3.3 Spatially-varying coefficients

We can further expand the family of supported PDEs by considering spatial variation of additional terms:

$$\begin{aligned} \nabla\left[\alpha(x)\nabla(u(x))\right] - \sigma(x)u(x) &= -f(x) \quad && x \in \Omega \\ u(x) &= g(x) && x \in \partial\Omega_d \\ \partial_n u(x) &= h(x) && x \in \partial\Omega_n\,, \end{aligned} \tag{5}$$

where $\alpha(x)$ and $\sigma(x)$ are spatially-varying diffusion and screening coefficients. In the special case of pure Dirichlet boundaries, Sawhney et al. [2022] show how the combination of a Girsanov transformation and an insight from null-collision random walks in heterogeneous volumes [Woodcock et al. 1965; Galtier et al. 2013] can turn Equation 5 into a screened Poisson equation with constant coefficients (Equation 1):

$$\begin{aligned} \Delta U(x) - \bar{\sigma}U(x) &= -(\bar{\sigma} - \sigma'(x))U(x) - f'(x), \quad && x \in \Omega, \\ U(x) &= \sqrt{\alpha(x)}g(x), && x \in \partial\Omega_d, \end{aligned} \tag{6}$$

where $f'(x) = f(x)/\sqrt{\alpha(x)}$, $\bar{\sigma}$ is a positive free parameter, and

$$\sigma'(x) = \frac{\sigma(x)}{\alpha(x)} + \frac{1}{2}\left(\frac{\Delta\alpha(x)}{\alpha(x)} - \frac{|\nabla(\ln\alpha(x))|^2}{2}\right). \tag{7}$$

The solution in terms of this modified problem is then given by $u(x) = \alpha(x)^{-1/2}U(x)$. A similar change is also possible in the more general case of mixed boundary conditions, please see the supplemental material for details. For brevity, we only discuss the Dirichlet version in the following.

Starting from Equation (6), we can build the WoS estimator using Equation (2). Omitting the source term $f(x)$, we have:

$$U(x) = \int_{\partial B(x)} P^{\bar{\sigma}}(x,x')\,U(x')\,\mathrm{d}x' + \int_{B(x)} G^{\bar{\sigma}}(x,y)(\bar{\sigma} - \sigma'(y))\,U(y)\,\mathrm{d}y. \tag{8}$$

Compared to the basic screened Poisson estimator, the volume integral over $B(x)$ now also depends on the solution $U(y)$. Since both integrals evaluate the solution recursively, Sawhney et al. [2022] probabilistically select one of the two terms in each iteration to prevent exponential runtime complexity. They derive the following selection probability using the relation between $P^{\bar{\sigma}}$ and $G^{\bar{\sigma}}$:

$$P^{\bar{\sigma}}(x,z) = 1 - \bar{\sigma}|G^{\bar{\sigma}}| = 1 - \bar{\sigma}\int_{B(x)} G^{\bar{\sigma}}(x,z)\,\mathrm{d}z. \tag{9}$$

Using this relation, the probability of sampling the volume term is $\mathbb{P}_{\partial B} = 1 - \mathbb{P}_B = 1 - \sigma|G^{\bar{\sigma}}|$. The surface and volume terms use the same strategies as the basic screened Poisson estimator. This yields the following estimator:

$$U(x_k) = \begin{cases} U(x_{k+1}) & \text{if } \xi < \mathbb{P}_{\partial B}, \\ U(x_{k+1})\left(1 - \frac{\sigma'(x_{k+1})}{\bar{\sigma}}\right) & \text{otherwise,} \end{cases} \tag{10}$$

where $\xi \in [0,1)$ is a uniform variate. The efficiency of this estimator depends on the choice of the free parameter $\bar{\sigma}$. Sawhney et al. [2022] set it to a global constant $\bar{\sigma} = \max_\Omega \sigma'(x) - \min_\Omega \sigma'(x)$, In Section 5.1, we further investigate the choice of $\bar{\sigma}$ and propose an acceleration data structure to set it adaptively.

However, even an adaptively chosen $\bar{\sigma}$ can produce samples with a very large weight, which adds significant variance especially in long random walks. Sawhney et al. [2022] adopt *weight windowing* to split the path once its throughput weight exceeds a given threshold, redistributing this weight across independently sampled path suffixes. We will shortly see that this complicates derivative computation, for which we present a solution in Section 4.1.

4 INVERSE SOLVER

With this background in place, we can now turn to inverse problems. Our goal is to solve a minimization problem of the form

$$\widehat{\boldsymbol{\pi}} = \arg\min_{\boldsymbol{\pi}} \ell(\mathbf{u}(\boldsymbol{\pi})) \tag{11}$$

using gradient-based optimization. The components of the vector $\mathbf{u}(\boldsymbol{\pi})$ represent discrete evaluations of the PDE's solution $u(x)$, e.g., on a regular grid. This solution is assumed to depend on a parameter vector $\boldsymbol{\pi}$ that controls the source term, boundary values, etc. The function ℓ is a differentiable loss function, for example a L_2 distance between $\mathbf{u}(\boldsymbol{\pi})$ and a reference. The optimization depends on our ability to differentiate both ℓ and the solution $\mathbf{u}(\boldsymbol{\pi})$, of which the latter is mainly of interest. For notational simplicity, we focus on a derivative $\partial_\pi := \partial/\partial\pi$ with respect to a single parameter π, but the resulting methods generalize to arbitrary parameter counts.

Fredholm integral equations. We begin by observing that all previously introduced estimators solve a Fredholm integral equation of the second kind:

$$u(x,\pi) = S(x,\pi) + \int_{\mathcal{Y}} K(x,x',\pi)\,u(x',\pi)\,\mathrm{d}x'. \tag{12}$$

This involves a sampling process on a domain $\mathcal{Y}$ involving accumulation of a source term S and recursive weighting by a kernel K. The source term S may itself be defined in terms of an integral but does not depend on the solution u. The key challenge we address in the following is the evaluation of the derivative $\partial_\pi u(x,\pi)$.

This formulation is reminiscent of the rendering equation [Kajiya 1986] solved by physically-based rendering algorithms:

$$L_o(\mathbf{x},\boldsymbol{\omega}) = L_e(\mathbf{x},\boldsymbol{\omega}) + \int_{S^2} f_s(\mathbf{x},\boldsymbol{\omega},\boldsymbol{\omega}')L_i(\mathbf{x},\boldsymbol{\omega}')\,\mathrm{d}\boldsymbol{\omega}'^{\perp}, \tag{13}$$

where L_o, L_i and L_e denote the outgoing, incident and emitted radiance and f is the *bidirectional scattering distribution function* (BSDF). In both equations, the solution appears in the integral on the right. Recursive sampling of this term by the associated estimators produces a random walk.

Derivative integrals. Recent work on differentiable rendering methods [Nimier-David et al. 2020; Vicini et al. 2021] casts differentiation into an independent random walk that propagates differential quantities (e.g., the derivative of radiance). Unlike conventional automatic differentiation, this does not require costly recording of the entire computational structure of the forward algorithm.

The principle underlying these methods can be observed by differentiating both sides of Equation 12 with respect to π. Under the assumption that π does not affect the position of discontinuities in the integrand or the integration domain $\mathcal{Y}$ itself, we can differentiate under the integral sign and apply the product rule, which produces the following expression for $\partial_\pi u(x,\pi)$:

$$\begin{aligned}\partial_\pi u(x,\pi) = \partial_\pi S(x,\pi) &+ \int_{\mathcal{Y}} \partial_\pi\left[K(x,x',\pi)\right]u(x',\pi)\,\mathrm{d}x' \\ &+ \int_{\mathcal{Y}} K(x,x',\pi)\,\partial_\pi u(x',\pi)\,\mathrm{d}x'.\end{aligned} \tag{14}$$

If we set $\hat{u}(x) := \partial_\pi u(x,\pi)$, then Equation (14) can be interpreted as a new Fredholm equation in terms of $\hat{u}(x)$ describing the propagation of the derivative. The first line defines the source term, while the second specifies recursive weighting by the original kernel K.

In contrast to the primal Equation (12), we now require *two* recursive estimates for $u(x)$ and $\hat{u}(x)$, which leads to quadratic time complexity as a function of path length. We instead use path replay backpropagation [Vicini et al. 2021] to solve this problem in linear time.

The following discussion assumes continuous functions S, K, and boundary values for now. The discontinuous case has received significant attention in rendering since it is needed for shape optimization, and some of the developed solutions such as *edge sampling* [Li et al. 2018] and *reparameterizations* [Bangaru et al. 2020; Zhang et al. 2020] can generalize to the Monte Carlo PDE setting. In Section 6.3, we showcase the optimization of a discontinuous domain boundary.

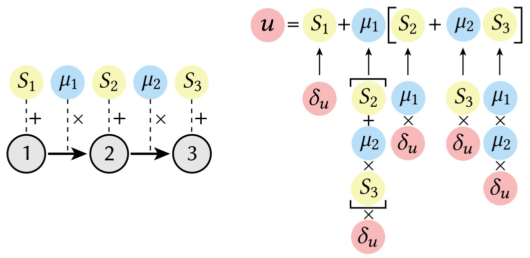
4.1 Path Replay Backpropagation

PRB is best explained using pseudocode. For this, let us first consider Algorithm 1, which computes a 1-sample estimate of the primal equation (12):

```
def solve(x):
    u = 0 # Running solution estimate
    β = 1 # Path throughput
    while not done:
        u += β * S(x, π)         # Accumulate source term
        x_new, pdf = sample_K(x) # Sample integral kernel
        μ = K(x, x_new, π) / pdf # Compute sampling weight
        β *= μ                   # Update throughput
        x = x_new
    return u
```

Algorithm 1. Monte Carlo solver for the Fredholm equation (12).

The running estimate u and throughput β are initially set to 0 and 1. At each step, the iteration accumulates throughput-weighted source contributions (e.g., the source term of the PDE or emission in



S_1 μ_1 S_2 μ_2 S_3
1 2 3
$u = S_1 + \mu_1 [S_2 + \mu_2 S_3]$
δ_u
(a) Random walk
(b) Contribution and backward derivative

Fig. 4. (**a**) Primal evaluation of a linear path with source (S) and weight (μ) terms. (**b**) Path Replay Backpropagation (PRB) [Vicini et al. 2021] replays the random walk once more to evaluate the backward derivative of all source and weight terms. The μ-derivative depends on the weighted source terms of the path *suffix*, which the method computes incrementally by subtracting the prefix from the total amount.

```
def solve_backward(x, $\delta_u$, u):
    $\beta$ = 1
    while not done:
        with ad_enabled(): # Gradients of "S"
            $\delta_\pi$ += backward($\beta$ * S(x,$\pi$) * $\delta_u$)

        # no gradient is tracked outside of ad_enabled()
        u -= $\beta$ * S(x,$\pi$)
        x_new, pdf = sample_next_step(x)
        v = u / K(x, x_new, $\pi$)
        with ad_enabled(): # Gradients of "K"
            $\delta_\pi$ += backward(v * K(x, x_new, $\pi$) * $\delta_u$)
        $\beta$ *= K(x, x_new, $\pi$) / pdf
        x = x_new
```

Algorithm 2. Path replay version of the Fredholm equation solver.

rendering) and then samples the next position x_new, while updating the throughput using a weight μ given by the ratio of K and the PDF. In rendering, this would be the ratio of the BSDF and the projected solid angle density of the sampling strategy. The evaluation of S may also involve a Monte Carlo estimator. If we were to use *next event estimation*, we would also consider it to be part of the source term S in this context.

PRB computes the reverse-mode derivative of this program. That is to say, if $\mathbf{J}$ is the Jacobian matrix of the primal PDE solver, then it evaluates the transpose matrix-vector product $\boldsymbol{\delta}_\pi = \mathbf{J}^T \boldsymbol{\delta}_u$ to convert a perturbation of the outputs ($\boldsymbol{\delta}_u$) into one of the inputs ($\boldsymbol{\delta}_\pi$).

To see what this entails, consider a short path with three vertices as illustrated in Figure 4a, where edges with weights μ_i scale later source contributions S_i. The reverse-mode derivative (Figure 4b) propagates the perturbation of the output δ_u to the terms of this weighted sum. The derivative of the source terms is given by $\delta_u \prod \mu_i$, which is easy to compute while building the path. The main challenge are the weights μ_i, whose derivative is proportional to the product of δ_u and the contribution of the *path suffix*. This information is not available while constructing the path, since it depends on vertices that have not yet been generated. One way to fix this issue is to generate the entire path first and store it for subsequent differentiation, but this leads to prohibitive storage costs especially on GPUs that will normally process millions of estimates in parallel.

PRB solves this issue by performing the same random walk twice, which it does by rewinding the underlying pseudorandom number generator to its earlier state and reproducing the same sequence of steps in a *replay phase*. This phase restarts at position $\mathbf{x}$ and takes a perturbation δ_u and the estimate u of the prior phase as input. It iteratively computes the necessary contribution of the path suffix by *subtracting* that of the prefix. This involves no approximations and removes the storage overheads.

Algorithm 2 realizes PRB for the previous example (Algorithm 1). The implementation evaluates the π-dependent functions S and K using *automatic differentiation* (AD), where backward() backpropagates gradients to δ_π. AD will usually eagerly differentiate the entire program, but here we actually want to localize its usage to a part of the loop body. Arithmetic outside the ad_enabled() blocks is therefore not tracked to avoid a costly AD computation graph spanning multiple loop iterations. Note that the above pseudocode evaluates certain terms twice (e.g., $S(x, \pi)$), which is purely for notational simplicity and can be avoided in practice.

Algorithm 2 realizes a *detached* sampling strategy [Zeltner et al. 2021; Vicini et al. 2021], which means that it estimates the parameter-dependent integral by only differentiating the integrand $f(x, \pi)$:

$$\int_X \partial_\pi f(x, \pi)\, \mathrm{d}x \approx \frac{\partial_\pi f(x_i, \pi)}{p(x_i, \pi)}, \tag{15}$$

where x_i is a sample with density $p(x_i, \pi)$. On the other hand, *attached* estimators furthermore differentiate the sampling strategy (i.e., how x_i is generated as a function of π). We use detached estimators throughout the paper but note that attached strategies could be useful to handle derivatives with respect to the domain boundary.

Algorithm 3 presents complete pseudocode of path replay for the WoS estimator of the screened Poisson equation based on the notation introduced in Section 3. The function detach(x) disconnects its argument from the automatic differentiation graph. This allows writing Algorithm 2 without duplicating the computation of μ once inside and once outside the ad_enabled() block. The supplemental material provides an extended version of this algorithm that also accounts for Neumann boundary conditions.

4.2 Weight windowing

Weight windowing is an area where differences between rendering and random walk-based PDE solvers become apparent. Assuming perfect sampling of K, the weight μ in a physically based renderer is upper-bounded[1] by 1—exceeding this value would imply that the material reflects more light than it receives. Non-electromagnetic radiation can sometimes require albedos that violate energy conservation (e.g., neutrons in nuclear reactor simulations), but this case is uncommon in rendering. As a consequence, the throughput β decays over time, enabling efficient stochastic path termination.

Random-walk based PDE solvers that target the spatially variable case do not have such an upper bound. The path weight β can become arbitrarily large, causing estimates with a long-tailed statistical distribution that impedes the optimization. Prior work [Sawhney

[1]This assumes that the renderer propagates *basic radiance*, the ratio of radiance and the squared index of refraction.

```
def wos_backward(x, u, δ_u, f, g, σ, σ̄, α):
    β = 1
    while True:
        R = distance_to_boundary(x)
        if R < ε:
            with ad_enabled():
                δ_π += backward(δ_u * β * g(x))
            return

        # Evaluate source contribution gradient
        y, pdf_green = sample_green(x, R, σ̄)
        with ad_enabled():
            S = β * f(y) * |G(x)| / sqrt(α(x) ∗ α(y))
            δ_π += backward(δ_u * S)
        u -= S

        # Decide sampling between boundary and volume.
        sample_volume = sampler.rand() < |G(x)| * σ̄
        if sample_volume:
            x_new = sample_green(x, R, σ̄)
        else:
            x_new = sample_boundary(x, R)

        # Evaluate throughput and its gradient
        with ad_enabled():
            μ = sqrt(α(x_new) / α(x))
            if sample_volume:
                σ‘ = eval_σ‘(σ(x_new), α(x_new),
                            ∇α(x_new), Δα(x_new))
                μ *= 1.0 - σ‘ / σ̄
            v = u / detach(μ)
            δ_π += backward(δ_u * μ * v)

        β *= μ
        x = x_new
```

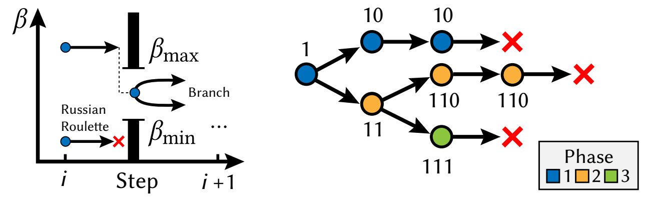
Algorithm 3. Differential Walk on Spheres for the general elliptic PDE.

et al. 2022] applied *weight windows* to address this problem, and we adopt this approach here as well.

The idea of a weight window is easily explained: if β ever exceeds a certain threshold (say, 2), we duplicate the current state and continue two independent walks starting from this position (Figure 5a). A high weight signals that the path prefix is inadequately sampled, and splitting it increases the computation budget retroactively for this part of the integration domain. We use a simple 2-way split[2] as opposed to a more complex n-way criterion [Sawhney et al. 2022] as we found this to be sufficient to address convergence issues in the spatially variable case. Finally, a standard Russian roulette stochastic stopping criterion is used on the lower end of the window. Concretely, our implementation uses $\beta_{\text{min}} = 1/2$ and $\beta_{\text{max}} = 2$.

A practical issue with branching walks is that their recursive implementation requires an unknown amount of stack memory. Conservative provisioning of a suitably large stack per GPU thread will generally exhaust the device memory, making this strategy impractical. We instead perform the computation in *phases*: each time, a random walk requires a split, we follow one branch in the current phase and atomically append the other to a global work queue for processing in the next phase. This process repeats until no work is left (Figure 5b). Memory usage then becomes proportional to the actual number of branches per phase, a significant improvement compared to conservative local provisioning of a stack.

[2]This means that states with $\beta \gg \beta_{\text{max}}$ will remain outside the window after branching.



(**a**) Branching & termination (**b**) Tree structure

Fig. 5. (**a**) Windowing of the path weight β to an interval is crucial to avoid statistical outliers in the spatially variable case. This involves either splitting or stochastically terminating paths that fall outside of an interval $(\beta_{\text{min}}, \beta_{\text{max}})$. (**b**) We partition the resulting branching random walk into a sequence of stack-less linear phases to enable efficient parallel execution of many random walks on the GPU. Each state is labeled by a bit-vector that identifies all branching decisions needed to reach it.

Finally, we associate a bit vector with every state that summarizes the branch decisions taken up to that point. The root of a random walk initializes this bit vector with 1, and every subsequent split either appends `0` or 1. The need for this will be clarified in Section 4.3.

4.3 Path replay for branching random walks

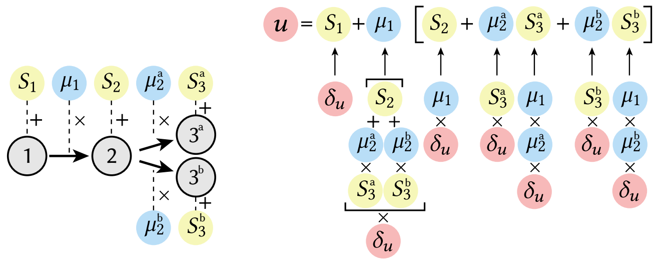
The PRB algorithm presented in Section 4.1 efficiently differentiates random walks with a linear structure, but it is unfortunately incompatible with the important weight windowing modification (Section 4.2) needed to handle the spatially variable case. We therefore extend Path Replay to branching random walks. The core mechanism of the method remains unchanged: it performs and subsequently replays random walks to efficiently evaluate the backward derivative. Our adaptation of this scheme to tree structures limits memory usage in exchange for redundant computation.

Section 4.2 explained how our solver processes linear chains in a parallel phase, using subsequent phases to handle splits. The high-level structure of this looks as follows:

```
1  while len(queue) > 0:
2      todo = []
3      for orig_state in queue:
4          final_state = solve(orig_state, write_branches_to=todo)
5      queue = todo
```

PRB would typically be applied to the combination of (`orig_state`, `final_state`) following line 4. However, this strategy will not work for a branching walk when `orig_state` refers to a split location from a prior phase. In this case, it is not sufficient to replay from this intermediate state—we must start *from the beginning* to generate derivative contributions linking prefix and suffix (Figure 6b).

We therefore store the initial configuration (position, pseudo-random number generator state) of the random walk along with the bit vector describing all branching decisions taken to reach the final state. Following this, we can reconstruct all derivative terms for a chain connecting root to leaf, which we additively combine with contributions from other parallel random walks and phases to compute a combined derivative.


S_1 μ_1 S_2 μ_2^a S_3^a
1 2 3^a 3^b
μ_2^b S_3^b
$u = S_1 + \mu_1 [S_2 + \mu_2^a S_3^a + \mu_2^b S_3^b]$
δ_u

(**a**) Branching random walk (**b**) Contribution and backward derivative

Fig. 6. (**a**) The spatially variable case depends random walks that can branch into separate suffixes. (**b**) The backward derivative of weights in the prefix preceding the split (μ_1) now involves quantities from both suffixes. Due to linearity, these don't all have to be computed at the same time. Our method performs replay per phase of the original primal computation (Figure 5b). In doing so, it only accumulates partial derivative terms associated with the subpath connecting roots to leaf nodes processed by the current phase.

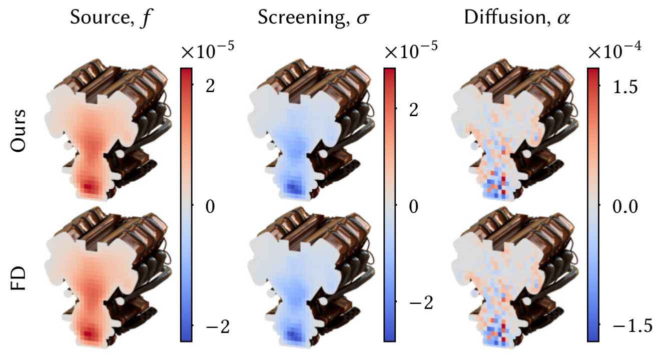


Fig. 7. Comparison between our gradient estimate and the finite differences reference (FD) for parameters of the general elliptic PDE (Eq. 5) in 3D.

The practicality of the redundant prefix traversal relies on the property that splitting is limited to outliers and doesn't happen excessively often along the same path. A variant on the other extreme of the storage-computation trade-off space could avoid redundant replay by allocating memory at every split to accumulate the weighted contribution of all suffixes originating at this position.

4.4 Validation and results

Validations. We first validate our gradient computations against finite differences. Figure 7 shows the parameter derivatives on a slice through the center of the domain. The coefficients are represented as cubically interpolated 3D textures of resolution 32^3. Their derivatives are computed by differentiating the L_2 norm of a 16^3 grid of solution evaluations in the domain. Gradients constructed using path replay and weight windowing reveal a perfect match against the reference. We provide similar validations for a 2D problem in the supplementary material.

Optimization results in 2D. Figure 8 shows the evolution of different optimizations that attempt to recover shown reference parameters by measuring the L_2 difference between a tentative solution and reference on a 32^2 grid. The parameters are represented as 16^2 textures with cubic interpolation. Each experiment optimizes one PDE coefficient, while keeping the others constant. This type of inversion is too ambiguous with just a single observation of the solution, and we generate many different configurations by prescribing linear combinations of sinusoidals and linear ramps to parameterize the temperature distribution on the boundary. In each iteration, we select 6 of them at random and use the Adam optimizer to take a step based on the summed gradient. We do not use any regularization but clip tentative parameters values to the range of the reference to improve robustness. In all cases, the difference between reference and tentative solution tends to zero towards the end of the optimization, producing a faithful parameter retrieval. We started all optimizations with a constant texture and did not find the results to be overly sensitive to initial values.

The normal derivative of the diffusion coefficient at Neumann boundaries must be zero, otherwise the problem transforms into a Robin boundary value problem, which we do not support. We only show Dirichlet boundaries in the diffusion coefficient optimization (last 2 rows) and will revisit the Neumann case in Section 6, where we demonstrate a practical application.

Optimization results in 3D. We demonstrate the generality of our method by performing similar optimizations in 3D. We designed an artificial heat conduction problem where we solve the general elliptic PDE (Equation 5) inside a model of an internal combustion engine. Although this setup does not directly correspond to a real world application, we believe that our method might be useful for practical heat conduction problems in the future. We used the same methodology as in the 2D optimizations (Figure 8). The solution tensor and the textures are defined on 32^3 and 16^3 grids, respectively. In the examples given in Figure 9, we can recover the reference coefficients almost perfectly.

Limitations. The presence of Neumann boundary generally results in longer random walks and higher variance. This increases optimization time due to the need to take more and smaller steps, and reduces the quality of reconstruction results. We analyze this effect by reconstructing the same high-frequency coefficients under Dirichlet and mixed boundary conditions. As shown in Figure 10, the size of the Neumann boundary considerably affects the quality of our reconstructions. To keep the comparison simple, we use the same number of samples across optimizations. This limitation is most pronounced when reconstructing the diffusion coefficients, as high frequency diffusion coefficients cause a high number of path splits. In the limit, large Neumann boundaries can therefore make the inverse solver impractical. We did not implement a 3D solver with Neumann boundaries, but expect similar limitations as in 2D.

We did not compare our method with state-of-the-art inverse FEM solvers, as we currently do not expect MC solvers to outperform these on our simple example problems. FEM solvers have received decades of research and our paper is only a first step in the use of inverse MC solvers. We believe that MC PDE solvers offer valuable flexibility which will lead to further progress on real inverse PDE applications.

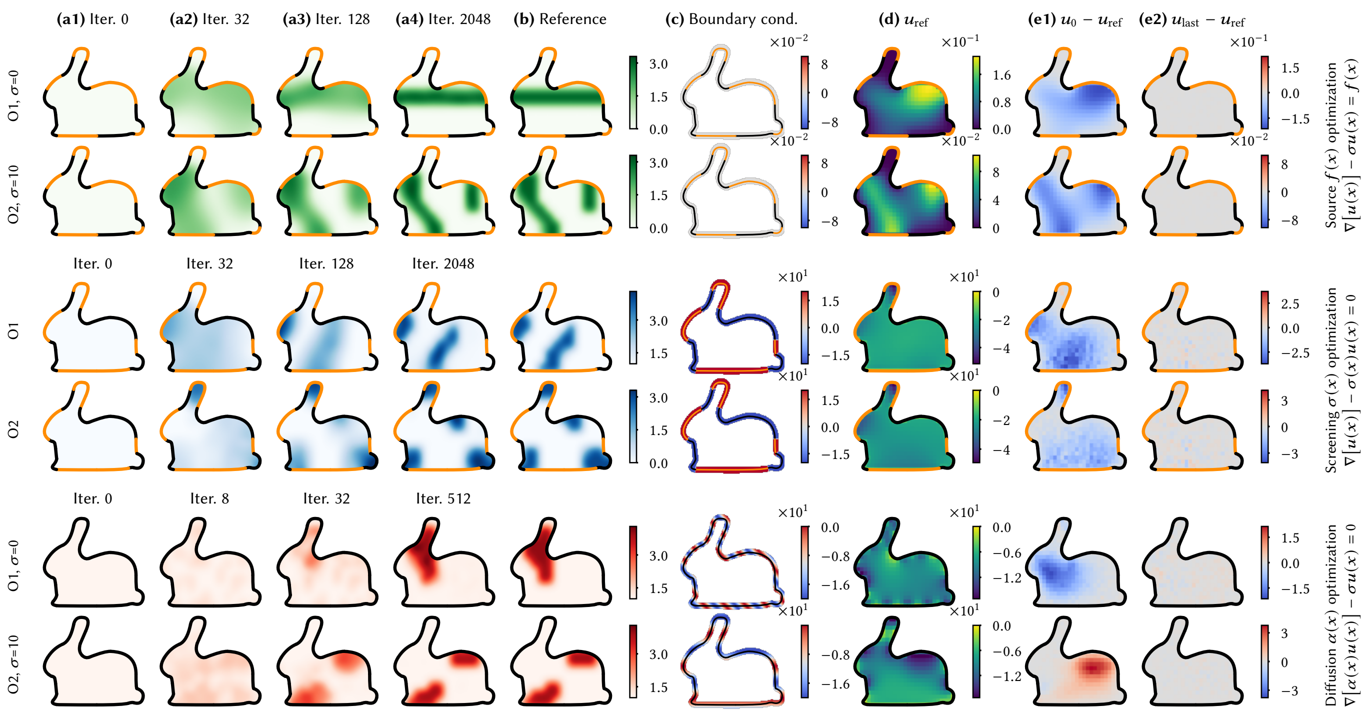


Fig. 8. Optimization results for different PDE coefficients in a 2D problem. Columns **(a)**: The evolution of the optimized parameter throughout the optimization. Column **(b)**: The reference. Columns **(c)** and **(d)**: one of the boundary conditions used in the optimization, and the corresponding reference solution inside the domain. Columns **(e)**: The difference between the reference solution in the first and the last iteration of the optimization. The orange boundary lines represent Neumann boundaries whereas dark lines represent Dirichlet boundaries.

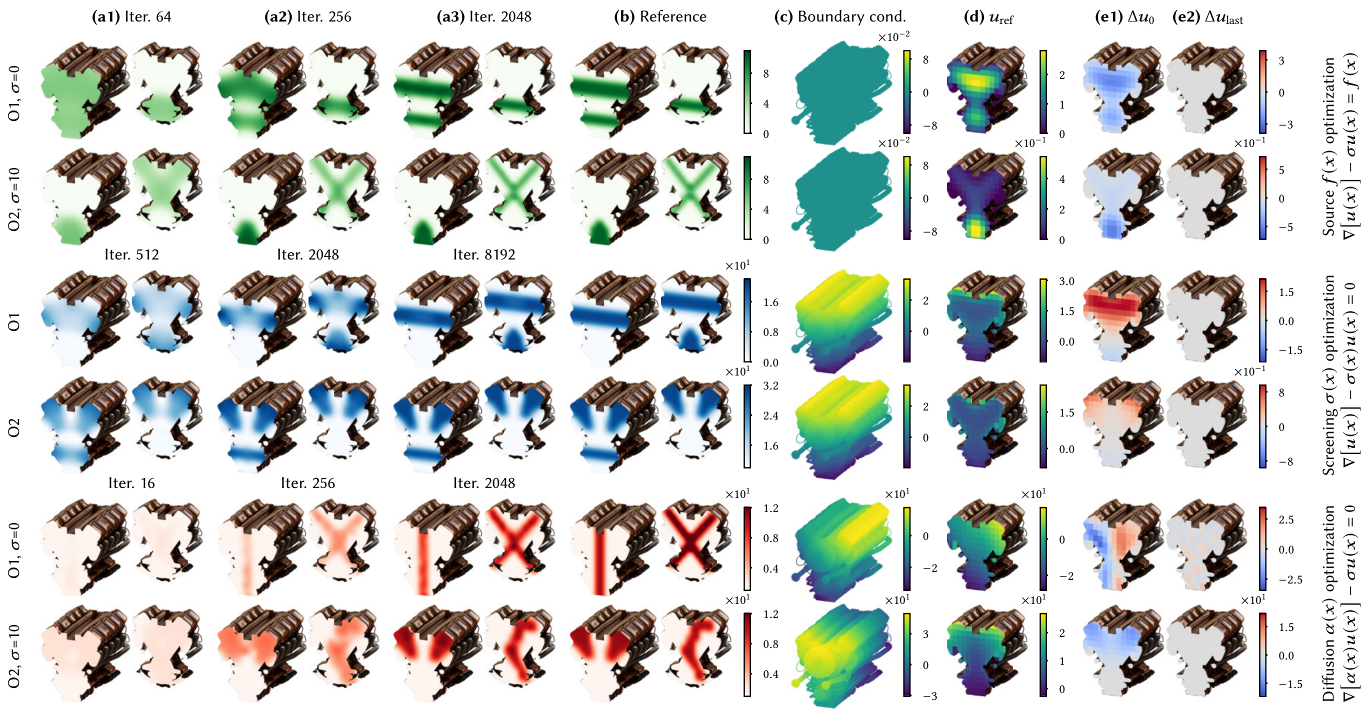

(a1) Iter. 64
(a2) Iter. 256
(a3) Iter. 2048
(b) Reference
(c) Boundary cond.
(d) u_{ref}
(e1) Δu_0
(e2) Δu_{last}
O1, σ=0
O2, σ=10
O1
O2
Iter. 512
Iter. 2048
Iter. 8192
Iter. 16
Iter. 256
Iter. 2048
Source $f(x)$ optimization $\nabla[u(x)] - \sigma u(x) = f(x)$
Screening $\sigma(x)$ optimization $\nabla[u(x)] - \sigma(x)u(x) = 0$
Diffusion $\alpha(x)$ optimization $\nabla[\alpha(x)u(x)] - \sigma u(x) = 0$

Fig. 9. Optimization results for different PDE coefficients in 3D. The result images are structured as in Figure 8. The values of reference and the optimized coefficients are visualized on 2 distinct slices whereas the the solution tensors (the last 3 rows) are visualized on a slice passing through the center of the model.

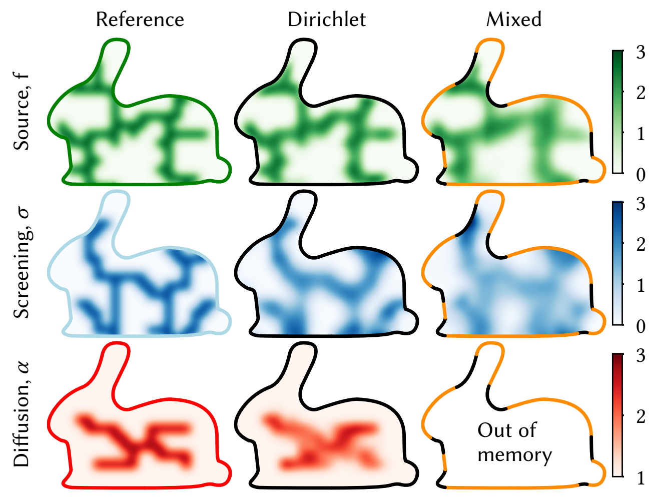


Fig. 10. Visualization of challenging optimizations with Dirichlet and mixed boundary conditions. **Left column**: The reference coefficient. **Center column**: The optimization result for the Dirichlet boundary value problem. **Right column**: The optimization result for the mixed boundary value problem. A large Neumann boundary can reduce the quality of the reconstruction and excessive path splits due to high-frequency diffusion variation may cause the solver to run out of memory (bottom right).

Performance. Table 2 provides the runtime performance for both primal and gradient computations for each optimization in Figures 8 and 9. We implemented all our methods using Dr.Jit [Jakob et al. 2022]. The reported timings are measured for the CUDA backend running on an NVIDIA RTX 4090.

Table 2. Average duration of different phases during each optimization step (in seconds). We report timings for computing the primal forward computation (P) and gradient computation (G). In parenthesis, we give the number of samples per point.

	2D		3D	
	P(2^{14})	G(2^{12})	P(2^{13})	G(2^{11})
Source O1	0.44	0.16	0.06	0.28
Source O2	0.35	0.12	0.05	0.15
Screening O1	1.83	0.99	0.36	0.37
Screening O2	1.02	0.59	0.25	0.26
Diffusion O1	0.73	0.87	0.70	2.98
Diffusion O2	0.51	0.32	0.48	1.00

5 ACCELERATING THE FORWARD SOLVER

One important limitation of the method as presented so far is that the diffusion coefficient must have a sufficiently low frequency. Noisy intermediate steps produced by the gradient-based optimization cause a high majorant $\bar{\sigma}$ (Section 3), which in turn slows down the optimizations. We now present an adaptive majorant and sphere radius selection to mitigate this effect. We also propose an optimized Green's function evaluation routine, that offers further speedups.

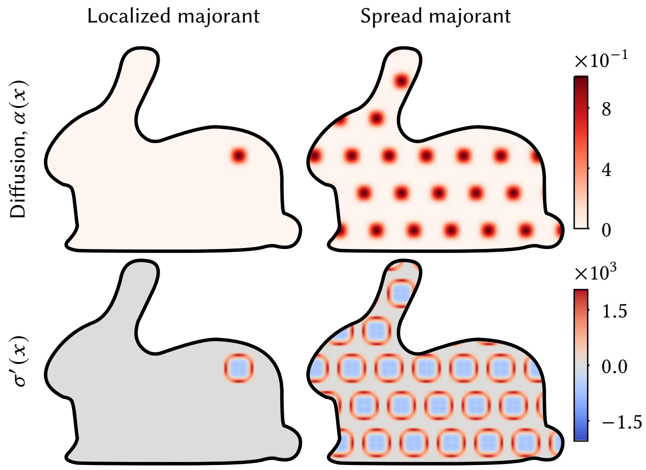


Fig. 11. **Left:** Even for a localized variation of the diffusion coefficient α, $\sigma'(x)$, and hence $\bar{\sigma}$, can become large. This reduces the expected per-step progress of the random walk, even if $|\sigma'(x)|$ is large only in a smaller subregion. **Right**: The value of $\bar{\sigma}$ and the expected step size remain the same if the variation is high across the domain.

5.1 Adaptive majorant and radius selection

We found that the solvers with spatially-varying coefficients can be accelerated by adaptively setting the majorant and picking a sphere radius that is *smaller* than the maximum admissible one. To explain why and how, we first recall the walk on spheres formulation given in Equation 8:

$$U(x) = \int_{\partial B(x)} P^{\bar{\sigma}}(x,x')U(x')\,\mathrm{d}x' + \int_{B(x)} G^{\bar{\sigma}}(x,x')(\bar{\sigma} - \sigma'(x'))U(x')\,\mathrm{d}x'.$$

Sawhney et al. [2022] globally set $\bar{\sigma}$ to $\max_\Omega \sigma'(x) - \min_\Omega \sigma'(x)$, which becomes large if the diffusion coefficient $\alpha(x)$ exhibits local high-frequency variation (see Figure 11).

A high global $\bar{\sigma}$ reduces the expected per-step progress of the random walk and increases computation time. In analogy to volume rendering, we characterize the average progress by the *mean free path*:

$$r_f(R,\bar{\sigma}) := \mathbb{E}[r|R,\bar{\sigma}] = \mathbb{P}_{\partial B}R + \mathbb{P}_B \int_0^R r\frac{G_r^{\bar{\sigma}}(r)}{|G^{\bar{\sigma}}|}\,\mathrm{d}r, \tag{16}$$

where $\mathbb{E}$ is the expected value operator, $r = \|x - x'\|$, R is the radius of the sphere, $G_r^{\bar{\sigma}}(r)$ is the radial distribution of the Green's function, $\mathbb{P}_{\partial B}$ and $\mathbb{P}_B$ are the selection probabilities of sampling the boundary and the volume integral, respectively. This definition is valid for the sampling routines outlined in Section 3.3.

Spatially-varying majorant. Inspired by similar approaches in volume rendering [Yue et al. 2010; Szirmay-Kalos et al. 2011], we increase the mean free path by replacing the global majorant by a spatially-varying one. This prevents local variations from slowing down convergence globally. We use a multi-resolution texture, that in each level stores the maximum value of $|\sigma'(x)|$ within a given ball $B(x,R)$. This is then used as a local majorant $\bar{\sigma}(x,R)$. That choice of majorant limits per-step throughput increases to at most a factor of 2 and thus prevents excessive splitting.

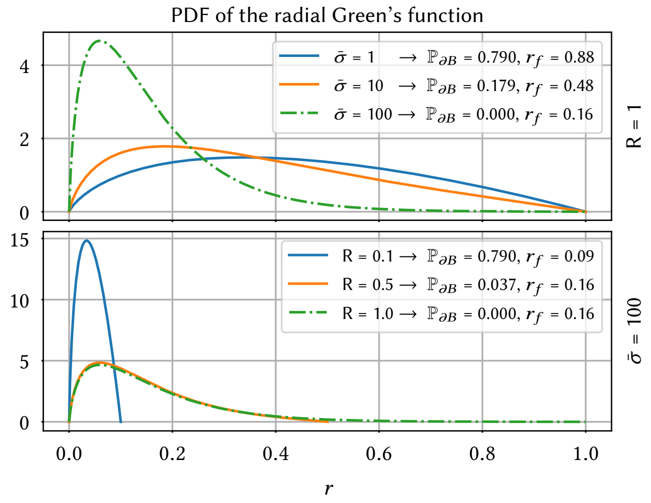


Fig. 12. Illustration of the radial Green's function and corresponding r_f and $\mathbb{P}_{\partial B}$ values for different parameters (in 2D). **Top:** For a fixed radius, $\mathbb{P}_{\partial B}$ and r_f decrease as $\bar{\sigma}$ increases. **Bottom:** When $\bar{\sigma}$ is kept constant, r_f plateaus above some critical radius value (green and orange lines).

Radius selection. We further increase the mean free path by adaptively selecting the radius of the sphere to integrate over. In Figure 12, we plot the Green's function for varying parameters R, $\bar{\sigma}$ and report the corresponding mean free path. For a fixed radius, increasing the majorant reduces $\mathbb{P}_{\partial B}$ and $G^{\bar{\sigma}}(r)$ gets peakier close to the origin, hence decreasing the mean free path. For a fixed majorant, increasing the radius initially increases the mean free path. However, the probability of sampling the boundary eventually converges to zero and the shape of the Green's function also approaches a limit behavior. Empirically, we found that increasing the radius beyond $R \approx 5/\sqrt{\bar{\sigma}}$ barely improves the mean free path.

Since $R\sqrt{\bar{\sigma}(x, R)}$ is a strictly increasing function in R, we can use bisection to find R_0 such that $R_0\sqrt{\bar{\sigma}(x, R_0)} \approx 5$. The value R_0 is an upper bound, and the optimal radius (in terms of mean free path) is a value between zero and R_0. We find the optimal value by evaluating $r_f(R, \bar{\sigma}(x, R))$ at equally spaced values of $R \in [0, R_0]$ using a precomputed lookup table.

Discussion. Figure 13 plots circles generated using our radius selection algorithm. If the center is far from a region with high-frequency variation, the circle is biggest circle excluding that region. For point very close to that region, the radius increases again as in any case the majorant is high. The algorithm still locally needs small steps, but our adaptive radius selection prevents this behavior from expanding to the entire domain. Figure 14 shows histograms of the path length distribution for both the baseline and our method. The average path length in our method is significantly lower in the case of localized high-frequency variation, without any noticeable difference in estimator variance. If the variation is high throughout the domain, there is no significant difference.

5.2 Efficient Sampling of the Green's Function in 2D

Solving PDEs with WoS and WoSt requires repeated sampling of the associated Green's function $G^{\sigma}(x, y)$. To sample proportional to

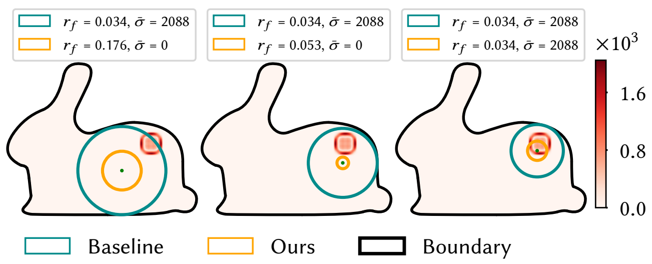

r_f = 0.034, $\bar{\sigma}$ = 2088
r_f = 0.176, $\bar{\sigma}$ = 0
r_f = 0.034, $\bar{\sigma}$ = 2088
r_f = 0.053, $\bar{\sigma}$ = 0
r_f = 0.034, $\bar{\sigma}$ = 2088
r_f = 0.034, $\bar{\sigma}$ = 2088
$\times 10^3$
1.6
0.8
0.0
Baseline
Ours
Boundary

Fig. 13. Illustration of our radius selection scheme (yellow) for the PDE defined in the left column of Figure 11. We compare to the baseline solver (blue), where the global majorant is set to $\max_\Omega |\sigma'(x)|$, and the radius is simply the minimum distance to the domain boundary. **Left:** If the query point is far from the region of high-frequency variation, our method selects the radius such that the sphere excludes the region where $|\sigma'(x)|$ is high (bright red values). **Middle:** As the query location moves closer, the radius shrinks accordingly. **Right:** Moving even closer, the radius increases again, as it is no longer efficient to try to exclude the high $|\sigma'(x)|$ values. We also provide the values of the mean free path r_f for both global and spatially-varying majorant. Our mean free path is up to 5× greater for this example.

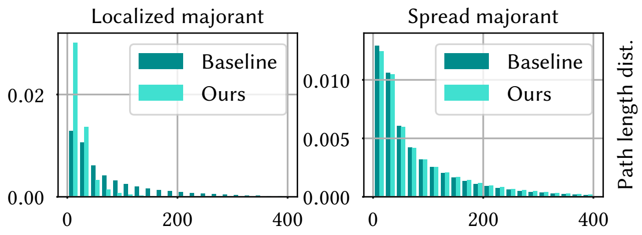


Fig. 14. Histograms over sampled path lengths for the global constant $\bar{\sigma}$ baseline and our adaptive method. This again uses the diffusion coefficients from Figure 11. On this particular example PDE, our method averages around 4× fewer interactions than the baseline for the localized variation (**left**). There is no speedup if the variation is high throughout the domain (**right**).

this function, we generate a random uniform direction, and sample a distance proportional to the radial Green's function $G_r^{\sigma}(r)$. First, we define $y = r/R$ and $z = R\sqrt{\sigma}$ to decrease the degrees of freedom in the normalized radial Green's function $G_r^{\sigma}(r)/R$, which we can now write as:

$$G(y, z) = y\left[K_0(yz) - I_0(yz)\frac{K_0(z)}{I_0(z)}\right] \tag{17}$$

where I and K refer to modified Bessel functions of the first and second kind (see supplementary material for details). Numerically robust evaluation of Bessel functions is relatively expensive. Given that we use simple boundary representations like quadratic Bézier curves, circles and SDFs, computation of the boundary queries are cheap. We found that evaluation of the Bessel functions is one of the main computational bottlenecks. Thus, we developed a sampling scheme that allowed us to run our simulations 5 to 10 times faster compared to the previous work which uses rejection sampling to sample the Green's function.

Our goal is to use the inverse transform method to draw y proportional to $G(y, z)$ for a given value of z, which requires inversion of the definite integral of this expression.

While this integral can indeed be determined and is given by

$$I_G(y,z) = \int_0^y G(y',z)\,\mathrm{d}y$$
$$= z^{-2}\left[1 - yzI_1(yz)\frac{K_0(z)}{I_0(z)} - yzK_1(yz))\right], \qquad (18)$$

its inversion requires numerical root finding as I_G^{-1} has no analytic form. This alone would be fine, but the definition (18) in terms of an extended set of Bessel functions makes this even harder than evaluating the original Green's function (17). We tried various ways of approximating $I_G(x,y)$ and transformed versions using polynomials but found that large numbers of coefficients were needed to obtain sufficiently good fits.

Near the origin, the modified Bessel function of the second kind resembles the negative logarithm, i.e., $K_0(z) \approx -\log z$ as $z \longrightarrow 0$ [DLMF 2024, 10.30.3], which turns out to be a defining feature that also influences the shape of I_G elsewhere. We incorporate this observation by adopting an approximation of the form

$$\tilde{I}_G(y,z) = P(y) + Q(y)\log y$$

where both P and Q are polynomials containing only positive even terms, i.e. $P(y) = p_0y^2 + p_1y^4 + \ldots$. This reduced the absolute error by more than two orders of magnitude compared ordinary polynomials at equal coefficient count. We fit $\tilde{I}_G$ to I_G by solving a linear system that enforces equality at Chebyshev nodes. The p_0 coefficient encodes the normalization constant. Given that our accelerated sphere tracing method requires to sample the Green's function in the domain $y \in [0,1]$ and $z \in [0,5]$, we used 6 coefficients per polynomial which creates a fit with maximum absolute error of $4 \cdot 10^{-6}$. This error is sufficient for our application as the rest of the solver is implemented in single precision.

To handle different parameters y, we perform many fits and represent the resulting coefficient table as a 1D texture. Importance sampling then performs a texture fetch to obtain coefficients for a given y followed by numerical inversion using Newton-bisection and the bracketing interval $[0,1]$. This requires the derivative

$$\tilde{I}'_G(x,y) = P'(x) + x^{-1}Q(x) + \log(x)\,Q'(x), \qquad (19)$$

which is cheap to evaluate in lockstep with I_G and also provides the needed sampling density. Finally, a good starting guess can greatly reduce the number of Newton-bisection steps. We use

$$x_0 = 1 - z^{-1}\cosh^{-1}(\cosh(z)(1-q)+q) \qquad (20)$$

where $q = \sqrt{\xi}$ for a random variate $\xi \in U[0,1)$. In the domain $y \in [0,1]$ and $z \in [0,5]$, rejection sampling is not very expensive and the main benefit of our method comes from the evaluation of the Green's function. But Green's function is extremely peaky for larger z and rejection sampling should be avoided.

Lastly, we emphasize that the Poisson kernel $P^\sigma(x,x')$ can be evaluated by $P^\sigma(x,x') = 1 - \sigma|G^\sigma|$ where $|G^\sigma|$ is the norm of the Green's function (see supplementary).

Most of the above steps are somewhat specific to the 2D case due to the complexity of the underlying Green's function. The 3D case only involves hyperbolic trigonometric functions, so the polynomial approximation is not needed in practice.

6 APPLICATION: ELECTRICAL IMPEDANCE TOMOGRAPHY

6.1 Problem statement

We apply our methods to electrical impedance tomography (EIT), which reveals an object's internal structure from measured electrical currents. This requires applying and measuring currents at electrodes placed on the object's surface, and then solving an inverse PDE problem to reconstruct the object's spatially-varying electrical conductance. Variations in conductance strongly correlate with specific internal structures. Since EIT measurements are on the domain's boundary and not in the interior, this problem is harder than the somewhat artificial setups studied so far (e.g., in Fig. 8 and 9).

A common application of EIT is non-invasive medical diagnosis, where it can aid detecting breast cancer, blood clots, and pulmonary emboli. Malignant tumors, with their high water content, exhibit conductivity levels around an order of magnitude higher than the surrounding healthy tissue [Adler and Boyle 2017]. Electrical impedance tomography also finds application geophysics (e.g., to locate mineral deposits) or material sciences (e.g., to identify corrosion and material defects) [Mueller and Siltanen 2012].

PDE formulation. In EIT, the following PDE governs the distribution of voltage $u(x)$:

$$\nabla\left[\alpha(x)\nabla u(x)\right] = 0, \quad x \in \Omega. \qquad (21)$$

The function $\alpha(x)$ is the spatially-varying conductance that we seek to reconstruct.

The choice of an *electrode model* determines the boundary conditions of the above PDE. We use a simple model that represents electrodes as infinitesimal points on the boundary and assumes negligible contact resistance. This results in the following Neumann boundary conditions:

$$\alpha(x_i)\frac{\partial u}{\partial n}(x_i) = I_i, \quad i = 1,\ldots,n, \qquad (22)$$

where x_i is the position of electrode i and I_i the injected current. At any other point on the domain boundary, the solution's normal derivative must be zero. The solution of this PDE is defined up to an additive constant. We can compute the missing offset by using that the sum of voltage values at the electrodes needs to be zero:

$$\sum_{i=0}^{n} u(x_i) = 0. \qquad (23)$$

Inverse problem. We reconstruct the conductance function by minimizing the difference between simulated voltage values and measured values at the electrodes using gradient descent. For the measurements, we inject positive current I at one electrode and "receive" the corresponding negative current $-I$ at another one. Measurements are then taken at all inactive electrodes, since we cannot measure the voltage at an electrode where current is injected. We activate electrodes in *skip-k* configurations, where between a pair of active electrodes, k electrodes are skipped.

6.2 Continuous conductance

Our Monte Carlo gradient estimators are flexible in their representation of the PDE's parameters. We begin by modeling the unknown

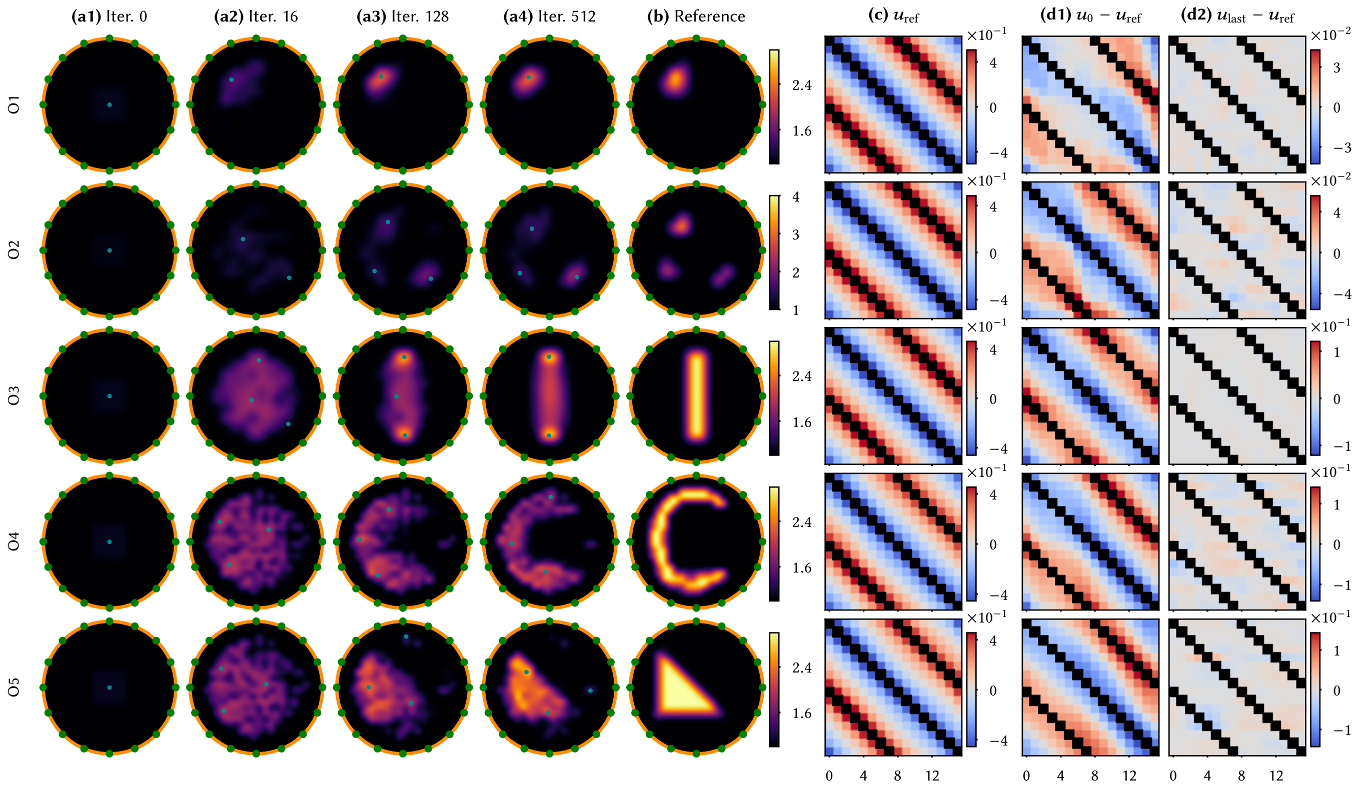


Fig. 15. Optimization results for continuous conductance EIT. Columns **(a)**: Evolution of the conductance coefficients. The blue dots inside the boundaries mark the circular Dirichlet boundaries (we add at most 3). Column **(b)**: The reference coefficients that we seek to recover. Column **(c)**: The reference voltage values at the electrodes. Each row of the voltage images is a distinct current injection pattern and each column is a different electrode. The currently active electrodes are marked by black pixels. Columns **(d)**: The difference to the reference voltages in the first and the last optimization iteration. The radius of the boundary is 0.22 and the radius of the Dirichlet boundaries is 0.00022. The resolution of the conductance texture is 16×16 for O1-O3, and 24×24 for O4 & O5.

conductance as a continuous function and present a discrete representation later.

Challenges. The forward EIT problem only has Neumann boundaries, and the absence of a screening term implies that no energy dissipates. Monte Carlo PDE solvers terminate either when reaching a Dirichlet boundary, or when, due to a screening term, the throughput decreases sufficiently to apply Russian roulette. Neither of this is the case here, which would lead to infinitely long random walks. This is similar to rendering a room containing only perfectly reflecting mirrors. In EIT, the path contributions become more and more low frequency as the number of interactions increases, eventually converging to a global constant.

We ensure a finite number of interactions by decreasing the throughput by a constant factor a user-defined threshold in each iteration, which results in a small bias in our estimators. Sawhney et al. [2023] achieve a similar effect by introducing a small screening coefficient if the walk becomes too long.

While EIT is a Neumann problem, we could add Dirichlet boundary conditions at any subset of points for which we know the actual voltage values. Building on this intuition, we place small, circular Dirichlet boundaries within the domain, allowing walks to terminate upon reaching them. In a preprocess, we estimate the voltage values at their location using our solver. While using these Dirichlet boundaries adds a small bias to our final estimator, they significantly reduce variance. The locations of the Dirichlet boundaries are determined by a clustering of high-conductance regions (see Section 7). Table 3 provides an analysis of the bias-variance tradeoff, demonstrating a substantial variance reduction without a noticeable difference in bias if the threshold path length is selected high.

Table 3. Bias, variance and computation time due to throughput reduction and added Dirichlet boundaries. k_d: number of Dirichlet boundaries, n_μ: threshold on walk length after which the throughput is decreased in every iteration, μ: throughput decrease factor. The number of samples per electrode for primal and gradient computation is 2^{15}, whereas 2^{23} to estimate the voltage at each Dirichlet boundary. The conductance function is the optimized conductance in the last iteration of O5 (Figure 15).

k_d	n_μ	μ	Bias $\times 10^3$	Var. $\times 10^4$	Primal (s)	Gradient (s)
0	150	0.99	2.04	25.4	0.19	4.32
0	50	0.90	11.8	4.34	0.05	0.27
3	∞	~	1.03	5.34	0.10	1.17
3	150	0.99	1.04	5.00	0.08	0.71
3	50	0.90	14.9	2.32	0.06	0.24

Table 4. Average duration of different phases during each optimization step (in seconds). From left to right, we report timings for computing the Dirichlet boundary voltages, primal forward computation and gradient computation. In parentheses, we give the number of samples per Dirichlet boundary (column 1) or electrode (columns 2, 3).

	Dirichlet boundaries (2^{23})	Primal (2^{20})	Gradient (2^{17})
O1	0.29	1.38	5.18
O2	2.34	1.23	3.63
O3	1.74	1.35	4.58
O4	3.65	2.03	11.23
O5	1.97	2.09	12.02

For future work, it could be interesting to combine this idea with de-biasing techniques [Misso et al. 2022].

Results. Figure 15 shows reconstruction results using a continuous conductance representation. The boundary conditions are generated with *skip-1*, *skip-3*, *skip-5* and *skip-7* current injection patterns. We compute reference voltage values using our forward solver, but minimize bias by disabling the throughput decrease and only using one Dirichlet boundary inside the domain.

The unknown conductance function $\alpha(x)$ is represented as a bicubically interpolated texture. We use an L_1 and a total variation regularizer to encourage sparsity and smoothness, respectively (each at a weight of 10^{-4}). Close to the boundary, we enforce a constant conductance to ensure that the normal derivative of the conductance is zero on the boundary. As detailed in the supplemental material, this prevents the Neumann boundary condition from turning into a more complex Robin boundary condition. After optimization, the voltage values computed from the optimized coefficients closely match the reference. Due to the ill-conditioned and challenging problem setup, it is no surprise that the reconstructed coefficients are not quite as accurate as for the simpler setups shown previously.

Table 4 shows a breakdown of the computation time of the different phases of our differentiable solver. We use a high number of Monte Carlo samples to compute the fixed voltage values of the circular Dirichlet boundaries. Nevertheless, the main computational bottleneck of our algorithm is the gradient computation, due to its repeated traversal of the branching random walk. This is more pronounced for the last two optimizations, as the higher frequency conductance results in more path splits.

6.3 Discrete conductance

Problem definition. We further apply the differentiable EIT solver to a discrete conductance representation. For this, we split the domain into two disjoint regions, each with a different constant conductance. Our aim is to reconstruct an interior object that is of significantly higher conductance than the surrounding background region. This object can therefore be modeled as a Dirichlet boundary with a constant value[3]. We illustrate this setup in Figure 16.

Gradient estimator. We estimate gradients for the resulting mixed-boundary problem using differentiable Walk on Stars. Unlike before, we differentiate the original estimator with respect to the *shape* of

[3]The constant is arbitrary, but can again be pinned after the solve using Equation 23.

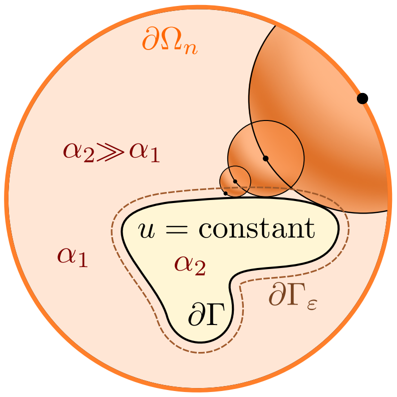


Fig. 16. Illustration of the discrete EIT setup. We model the high-conductance inner object as a Dirichlet boundary, since its conductance α_2 is much larger than the background conductance α_1. The random walks sampled by WoSt reflect on the outer Neumann boundary and terminate when they reach the inner object.

the inner Dirichlet boundary $\partial\Gamma$. For a point x where the closest boundary is $\partial\Gamma$, we need to differentiate a Walk on Spheres step:

$$\partial_\pi u(x,\pi) = \partial_\pi \int_{\partial B(x)} \frac{1}{|\partial B(x)|}(\mathbb{1}_{\Omega\setminus\partial\Gamma_\varepsilon}(x)u(x,\pi) + \mathbb{1}_{\partial\Gamma_\varepsilon}(x)g(x))\,\mathrm{d}x,$$

where $\mathbb{1}_A(x)$ is an indicator function that is 1 if $x \in A$ and 0 otherwise. This formulation makes the termination of walks reaching the epsilon shell $\partial\Gamma_\varepsilon$ explicit. The discontinuities due to the indicator function could be handled using a reparameterization [Loubet et al. 2019] or explicit sampling [Li et al. 2018]. However, since ε is small, we found a simpler approach to work well: Whenever a path terminates on a Dirichlet boundary, we compute a solution derivative $\partial_\pi u(x,\pi)$ using a first-order expansion along the normal direction:

$$u(x,\pi) \approx V + d(x,\pi)\partial_n u(x_n,\pi), \tag{24}$$

where $d(x,\pi)$ is the boundary distance, V the constant Dirichlet voltage of the inner object and x_n the closest boundary point (see Figure 17a). On the boundary, the spatial solution derivative is continuous and colinear to the normal of the Dirichlet boundary (Figure 17b). We use this local expansion to create the following

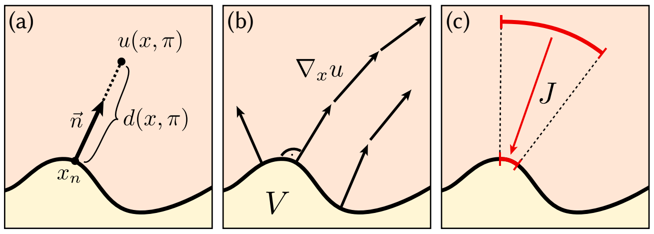


Fig. 17. **(a)** We locally approximate the solution value at a distance $d(x,\pi)$. **(b)** The spatial gradient of the solution is perpendicular to the surface and has no tangential component. **(c)** The Jacobian term corrects for the density change due to surface curvature.

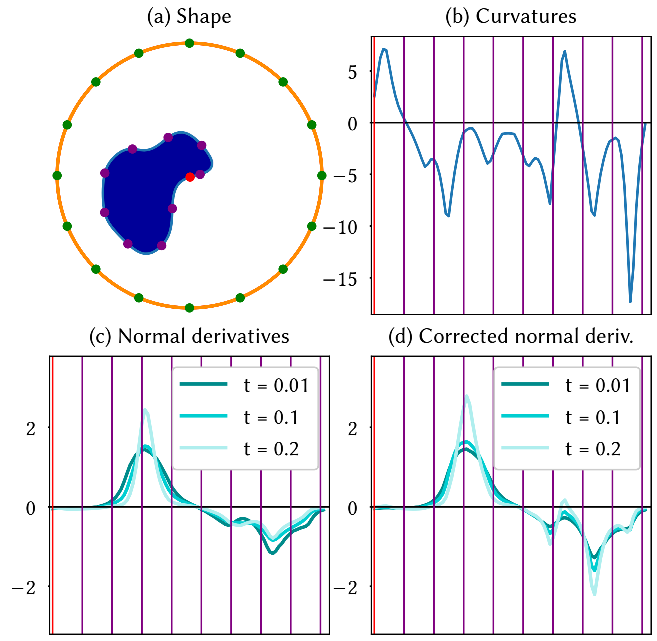


Fig. 18. Illustration of the bias due to evaluating the normal derivative at an offset. **(a)** The used example setup, where red and purple points correspond to the vertical lines in other plots (in clockwise order), and the green points are the electrodes. The radius of the outer boundary (yellow) is 1. **(b)** Curvature at the boundary. **(c)** The normal derivative of the solution computed at different offsets. **(d)** Applying the Jacobian correction factor reduces the bias due to an increase in offset distance.

gradient estimator:

$$\partial_\pi u(x_k, \pi) = \begin{cases} \partial_\pi V + \partial_\pi d(x_k, \pi)\partial_n u(x_{k,n}, \pi) & x_k \in \partial\Gamma_\varepsilon \\ \partial_\pi u(x_{k+1}, \pi) & \text{otherwise,} \end{cases} \tag{25}$$

where $x_{k,n}$ is the projection of x_k onto the object boundary. We do not differentiate the normal derivative with respect to the parameters and found the bias due to this approximation to be negligible.

Normal derivative. Evaluating the normal derivative very close to the boundary is both noisy and biased due to the proximity to the epsilon shell [Sawhney and Crane 2020]. We therefore evaluate the normal derivative $\partial_n u(x, \pi)$ at a fixed distance t from the Dirichlet boundary. As the spatial gradient of the solution $(\nabla_x u)$ is divergence-free in our setup, we correct magnitude of the normal derivative $(\partial_n u)$ by the Jacobian of this transformation (Figure 17c):

$$\partial_n u(x_n, \pi) \approx |J(x_n, x_t)| \partial_n u(x_t, \pi), \tag{26}$$

where x_t is the offset location, $|J(x_n, x_t)| = 1 - t\rho(x_n)$ and $\rho(x_n)$ the curvature at the boundary point. The choice of t exposes a bias-variance tradeoff. Figure 18 illustrates the effect of t on the estimated normal derivative.

Gradient validation. We validate our gradients against finite differences in Figure 19. We model the inner shape using an signed distance function (SDF) represented by a bicubically interpolated grid. As expected, the estimator bias increases if we use a larger normal derivative offset t. We found $t = 10^{-2}$ to closely match finite differences and provide a practical bias-variance tradeoff.

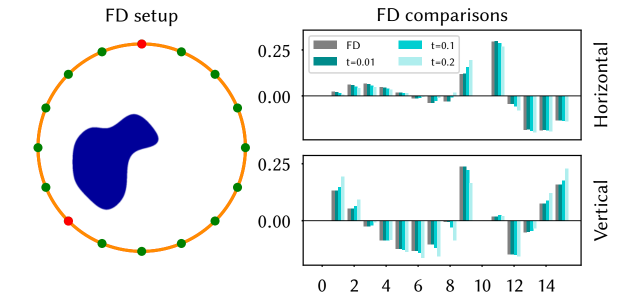


Fig. 19. Validation of our derivative estimator against finite differences (FD). We compute derivatives of the solution at the electrodes with respect to vertical and horizontal translations of the inner shape. Starting with 0 at the top, electrodes are numbered in clockwise order. Injected current locations are marked in red (electrodes 0, 10) and the radius of the boundary is 1.

Optimization results. We show optimization results for synthetic example problems in Figure 20. The discrete representation reduces ambiguities compared to the continuous case. We noticed that concave regions are more difficult to reconstruct (e.g., result O4), since the local voltage values become very similar to the voltage of the Dirichlet boundary.

Finally, we apply our method to measured data of metal conductors in a saline solution, using the dataset by Hauptmann et al. [2018]. The dataset is created using the KIT4 (Kuopio Impedance Tomography) measurement system [Hauptmann et al. 2017]. The setup is vertically uniform so that it can be modeled on a 2D domain. We show reconstructions using the SDF representation in Figure 21 and, as an alternative, using a parameterized circle in Figure 1. Both SDF and circle parameterization allows reconstructing the shape and position of the metal conductor. The reconstructions are not perfect, which is likely due to the approximations taken by the electrode model. Further improvements could be made by using more realistic electrode models or formulating the problem with Robin boundary conditions [Maire and Simon 2015; Miller et al. 2024].

7 IMPLEMENTATION DETAILS

We now briefly discuss some key implementation details.

Evaluation of boundary conditions. For all optimization results, we compute gradients using multiple different boundary conditions. Since the boundary values do not affect sampling, we improve efficiency by simultaneously computing the solution for six different boundary conditions (instead of running independent solves).

Circular Dirichlet boundaries. In each optimization iteration for EIT, we find the circular Dirichlet boundaries by using K-means clustering on the value and norm of the conductance function. We eliminate clusters that are too close to each other and pick the top three clusters according to the average conductance over their respective region. We then estimate the voltage at each selected location and use the result during gradient computation. This choice

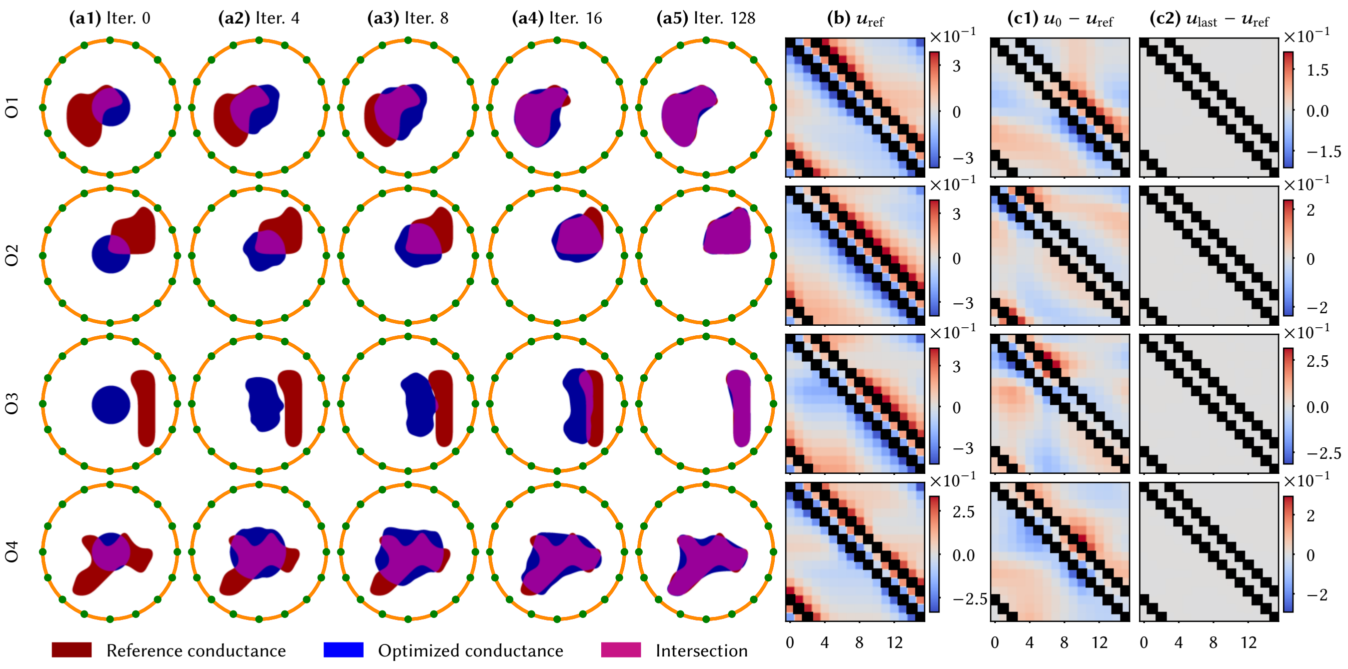


Fig. 20. Optimization results using the discrete conductance model on synthetic data. Columns **(a)**: Evolution of the coefficients over gradient descent iterations. Column **(b)**: The reference voltage values on the electrodes, with each row representing a different current injection pattern. The nodes at which current is injected are black. Columns **(c)** show differences in voltage values for first and last iteration. The radius of the outer boundary is 0.14.

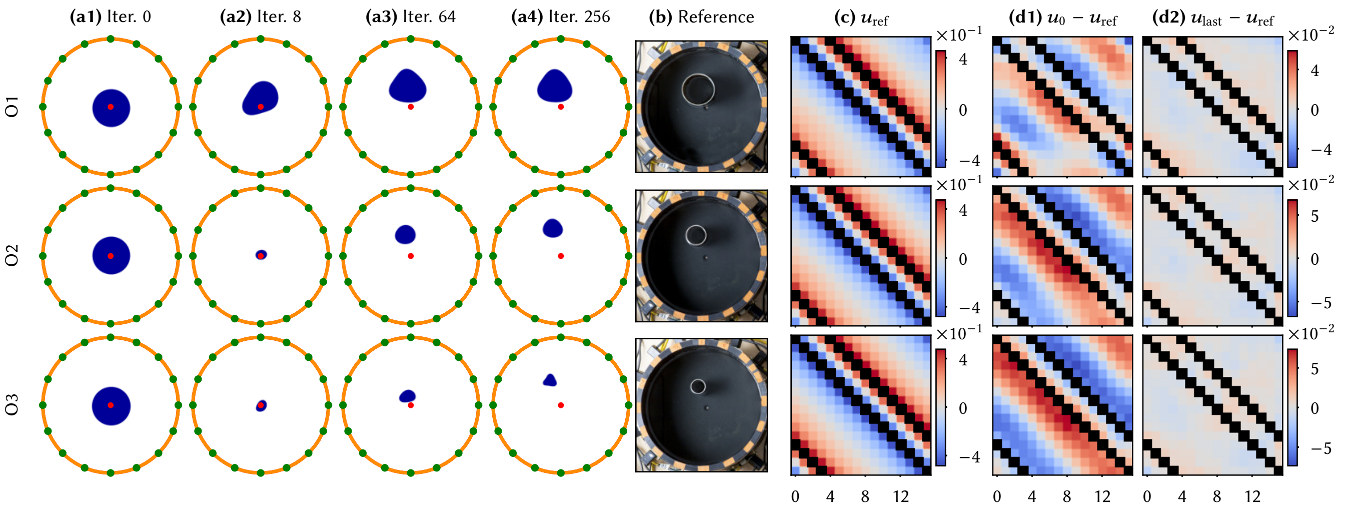


Fig. 21. Optimization results on the measured EIT data using the SDF representation. We use the same color scheme and layout as in Figure 20.

of Dirichlet boundary locations is physically-motivated: The Dirichlet boundaries can be understood as regions of infinite conductance. Therefore, placing them in high conductance regions reduces bias.

Normal derivatives. The shape derivatives in discrete EIT require estimating the solution's normal derivative. Even when evaluated at an offset way from the boundary, the normal derivative is noisy. We thus precompute it for each texel of the SDF using $\approx 2^{14}$ samples. For the circle representation (Figure 1), we precompute the normal derivative for an evenly distributed set of points on the circle itself.

Handling interpolated SDFs. The distance values in interpolated SDFs can be inaccurate, especially at low resolutions (see Figure 22). Our discrete EIT gradients rely on an accurate, differentiable distance evaluation. We therefore correct for inaccuracies by dividing the distance by the norm of the spatial gradient of the SDF when evaluating Equation 25. Similar issues with interpolated SDFs were observed in differentiable rendering [Vicini et al. 2022].

Redistancing. To ensure that a texture represents a valid signed distance function, we apply redistancing after each optimization step.

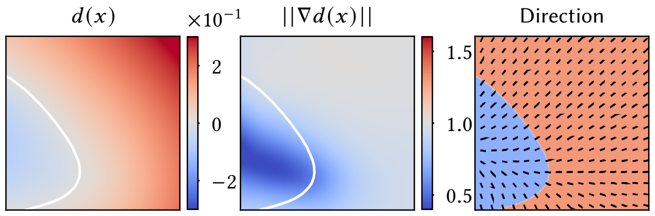


Fig. 22. **Left:** Distance values of a grid-based SDF. We mark the zero-level set using white lines. **Center:** The norm of the spatial gradient differs from 1, violating the Eikonal equation $\|\nabla d(x)\| = 1$. In the blue region, the evaluated SDF values diverge from the true geometric distance. **Right:** The orientation of the spatial gradient $\nabla d(x)$ is correct, which implies that simply re-normalizing largely corrects interpolation issues.

We do this using the *scikit-fmm* [Furtney 2023] library. It does not account for the bicubic interpolation that we use during simulation. To compensate for this, we upsample the SDF to a resolution of 1024^2 using bicubic interpolation, then perform the redistancing and finally downsample back to the original resolution.

8 CONCLUSION

We introduced practical derivative estimators for Walk on Spheres and various extensions thereof. Differentiable Monte Carlo methods admit a great flexibility in the underlying representation and set of optimized parameters. For practical applications, this could allow to not only reconstruct the main parameters of interest, but also simultaneously calibrate measurement setups In medical EIT, determining electrode locations or body shape of patients is still an active research area. Unlike MC solvers, FEM-based methods would have to run a costly domain remeshing step in each iteration. Another benefit of MC PDE solvers is their support for a wide range of parameter representations. The shape of the boundary, boundary conditions, or PDE coefficients can be parameterized by various functions, enabling the use of complex, domain-specific priors. We demonstrate this for EIT, where we use circles and SDFs for discrete shape representations.

The main limitation of our method is the performance and generality of the forward solvers. For example, high-frequency spatial variation in the diffusion coefficient requires a large number of costly path splits. These problems are exacerbated for pure Neumann problems without screening term. The effect of parameters on the final PDE solution is often the result of very long random walks, which means that, unlike in rendering, we cannot reduce the number of interactions to a fixed maximum.

We believe that there are many interesting future directions to explore. Our application to EIT only scratches the surface of potential practical usecases. We expect novel applications to come with their own set of challenges, but hope that our methods provide a useful foundation. The similarities with light transport simulation and inverse rendering mean that their solution can benefit from the computer graphics community.

9 ACKNOWLEDGEMENTS

We thank Rohan Sawhney and Dario Seyb for their valuable clarifications on the forward solvers.

REFERENCES

Andy Adler, John H Arnold, Richard Bayford, Andrea Borsic, Brian Brown, Paul Dixon, Theo JC Faes, Inéz Frerichs, Hervé Gagnon, Yvo Gärber, et al. 2009. GREIT: a unified approach to 2D linear EIT reconstruction of lung images. *Physiological measurement* 30, 6 (2009).

Andy Adler and Alistair Boyle. 2017. Electrical Impedance Tomography: Tissue Properties to Image Measures. *IEEE Transactions on Biomedical Engineering* 64, 11 (2017).

Pieter Bakbouk, Ghada and Peers. 2023. Mean Value Caching for Walk on Spheres. In *Proceedings of EGSR.*

Sai Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39, 6 (2020).

David C Barber and Brian H Brown. 1984. Applied potential tomography. *Journal of Physics E: Scientific Instruments* 17, 9 (1984), 723.

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. (Proc. SIGGRAPH)* 42, 4 (2023), 1–20.

Richard H Bayford. 2006. Bioimpedance tomography (electrical impedance tomography). *Annu. Rev. Biomed. Eng.* 8 (2006), 63–91.

M.Z Brainina, V.L Generozov, V.G Kuznetsov, and V.A Sakovich. 1967. Evaluation of dose derivatives by the Monte Carlo method for optimizing protective screen shape and composition. *U. S. S. R. Comput. Math. and Math. Phys.* 7, 4 (1967), 335–340.

Alberto P Calderón. 2006. On an inverse boundary value problem. *Computational & Applied Mathematics* 25 (2006), 133–138.

J.R Cannon. 1964. Determination of certain parameters in heat conduction problems. *J. Math. Anal. Appl.* 8, 2 (1964), 188–201.

Margaret Cheney, David Isaacson, and Jonathan C. Newell. 1999. Electrical Impedance Tomography. *SIAM Rev.* 41, 1 (1999), 85–101.

Margaret Cheney, David Isaacson, Jonathan C Newell, S Simske, and J Goble. 1990. NOSER: An algorithm for solving the inverse conductivity problem. *International Journal of Imaging systems and technology* 2, 2 (1990), 66–75.

DLMF 2024. *NIST Digital Library of Mathematical Functions.* https://dlmf.nist.gov/, Release 1.2.0 of 2024-03-15. https://dlmf.nist.gov/ F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds..

Francis H.R. França, John R. Howell, Ofodike A. Ezekoye, and Juan Carlos Morales. 2003. Inverse design of thermal systems with dominant radiative transfer. Advances in Heat Transfer, Vol. 36. Elsevier.

Jason Furtney. 2023. *scikit-fmm.* https://github.com/scikit-fmm/scikit-fmm

Mathieu Galtier, Stéphane Blanco, Cyril Caliot, Christophe Coustet, Jérémi Dauchet, Mouna El Hafi, Vincent Eymet, Richard Fournier, Jacques Gautrais, Anaïs Khuong, et al. 2013. Integral formulation of null-collision Monte Carlo algorithms. *Journal of Quantitative Spectroscopy and Radiative Transfer* 125 (2013).

Ioannis Gkioulekas, Anat Levin, and Todd Zickler. 2016. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *European Conference on Computer Vision (ECCV).* Springer International Publishing, 685–701.

Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. 2013. Inverse Volume Rendering with Material Dictionaries. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 32, 6, Article 162 (Nov. 2013).

R. Guardo, C. Boulay, and M. Bertrand. 1991. A Neural Network Approach To Image Reconstruction In Electrical Impedance Tomography. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society Volume 13: 1991.* 14–15.

Sarah J Hamilton, CNL Herrera, JL Mueller, and Alan Von Herrmann. 2012. A direct D-bar reconstruction algorithm for recovering a complex conductivity in 2D. *Inverse problems* 28, 9 (2012).

Sarah J Hamilton, JL Mueller, and TR Santos. 2018. Robust computation in 2D absolute EIT (a-EIT) using D-bar methods with the 'exp'approximation. *Physiological Measurement* 39, 6 (2018).

Andreas Hauptmann, Ville Kolehmainen, Nguyet Minh Mach, Tuomo Savolainen, Aku Seppänen, and Samuli Siltanen. 2017. Open 2D electrical impedance tomography data archive. (2017). arXiv:1704.01178

Andreas Hauptmann, Ville Kolehmainen, Nguyet Minh Mach, Tuomo Savolainen, Aku Seppänen, and Samuli Siltanen. 2018. *2D electrical impedance tomography dataset.*

Wenzel Jakob, Sébastien, Nicolas Roussel, and Delio Vicini. 2022. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 41, 4 (July 2022), 124:1–124:19.

James T Kajiya. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques.* 143–150.

Pramook Khungurn, Daniel Schroeder, Shuang Zhao, Kavita Bala, and Steve Marschner. 2015. Matching Real Fabrics with Micro-Appearance Models. *ACM Trans. Graph.* 35, 1, Article 1 (Dec. 2015).

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37, 6 (2018), 222:1–222:11.

Xiuyan Li, Yang Lu, Jianming Wang, Xin Dang, Qi Wang, Xiaojie Duan, and Yukuan Sun. 2017. An image reconstruction framework based on deep neural network for electrical impedance tomography. In *IEEE International Conference on Image Processing (ICIP)*. 3585–3589.

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 *Proc. SIGGRAPH Asia (Conference track)*. Article 34.

Dong Liu and Jiangfeng Du. 2021. Shape and topology optimization in electrical impedance tomography via moving morphable components method. *Structural and Multidisciplinary Optimization* 64, 2 (2021), 585–598.

Dong Liu, Anil Kumar Khambampati, and Jiangfeng Du. 2017. A parametric level set method for electrical impedance tomography. *IEEE Trans. med. imaging* 37, 2 (2017).

Shengheng Liu, Ruisong Cao, Yongming Huang, Taweechai Ouypornkochagorn, and Jiabin Jia. 2020a. Time sequence learning for electrical impedance tomography using Bayesian spatiotemporal priors. *IEEE Transactions on Instrumentation and Measurement* 69, 9 (2020), 6045–6057.

Shengheng Liu, Yongming Huang, Hancong Wu, Chao Tan, and Jiabin Jia. 2020b. Efficient multitask structure-aware sparse Bayesian learning for frequency-difference electrical impedance tomography. *IEEE Trans. industrial informatics* 17, 1 (2020).

Shengheng Liu, Jiabin Jia, Yimin D Zhang, and Yunjie Yang. 2018. Image reconstruction in electrical impedance tomography based on structure-aware sparse Bayesian learning. *IEEE transactions on medical imaging* 37, 9 (2018), 2090–2102.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 38, 6 (Dec. 2019).

Iván Lux and Lázló Koblinger. 1990. *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*. CRC Press, Boston.

Sylvain Maire and Martin Simon. 2015. A partially reflecting random walk on spheres algorithm for electrical impedance tomography. *J. Comput. Phys.* 303 (2015).

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, Vol. 23. ACM.

Guennady A. Mikhailov. 1967. Monte-Carlo calculation of derivatives of functionals from the solution of the transfer equation according to the parameters of the system. *U. S. S. R. Comput. Math. and Math. Phys.* 7, 4 (1967), 274–281.

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2023. Boundary Value Caching for Walk on Spheres. *ACM Trans. Graph. (Proc. SIGGRAPH)* 42, 4, Article 82 (July 2023).

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

Zackary Misso, Benedikt Bitterli, Iliyan Georgiev, and Wojciech Jarosz. 2022. Unbiased and consistent rendering using biased estimators. *ACM Trans. Graph. (Proc. SIGGRAPH)* 41, 4 (July 2022).

Sean Molesky, Zin Lin, Alexander Y. Piggott, Weiliang Jin, Jelena Vucković, and Alejandro W. Rodriguez. 2018. Inverse design in nanophotonics. *Nature Photonics* 12, 11 (01 Nov 2018), 659–670.

Lukas Mosser, Olivier Dubrule, and Martin J. Blunt. 2020. Stochastic Seismic Waveform Inversion Using Generative Adversarial Networks as a Geological Prior. *Mathematical Geosciences* 52, 1 (2020).

Jennifer L Mueller and Samuli Siltanen. 2012. *Linear and nonlinear inverse problems with practical applications*. SIAM.

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 Trans. Graph. (Proc. SIGGRAPH)* 40, 4, Article 97 (July 2021).

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4 (July 2020).

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 38, 6 (Nov. 2019).

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 (July 2013), 101–108.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically Based Rendering: From Theory to Implementation* (4th ed.). The MIT Press.

Yang Qi, Dario Seyb, Benedikt Bitterli, and Wojciech Jarosz. 2022. A bidirectional formulation for Walk on Spheres. *Comp. Graph. Forum (Proc. EGSR)* 41, 4 (2022).

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. (Proc. SIGGRAPH Asia)* 41, 6, Article 240 (Nov. 2022).

Fadil Santosa and Michael Vogelius. 1990. A backprojection algorithm for electrical impedance imaging. *SIAM J. Appl. Math.* 50, 1 (1990), 216–243.

Rohan Sawhney and Keenan Crane. 2020. Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4 (2020).

Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. 2022. Grid-free Monte Carlo for PDEs with spatially varying coefficients. *ACM Trans. Graph. (Proc. SIGGRAPH)* 41, 4 (2022).

Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. 2023. Walk on Stars: A Grid-Free Monte Carlo Method for PDEs with Neumann Boundary Conditions. *ACM Trans. Graph. (Proc. SIGGRAPH)* (2023).

László Szirmay-Kalos, Balázs Tóth, and Milán Magdics. 2011. Free Path Sampling in High Resolution Inhomogeneous Participating Media. *Comp. Graph. Forum* 30, 1 (2011), 85–97.

Zdravko Velinov, Marios Papas, Derek Bradley, Paulo Gotardo, Parsa Mirdehghan, Steve Marschner, Jan Novák, and Thabo Beeler. 2018. Appearance Capture and Modeling of Human Teeth. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37, 6 (Dec. 2018), 207:1–207:13.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path Replay Backpropagation: Differentiating Light Paths using Constant Memory and Linear Time. *ACM Trans. Graph. (Proc. SIGGRAPH)* 40, 4 (Aug. 2021).

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 41, 4 (July 2022).

E Woodcock, T Murphy, P Hemmings, and S Longworth. 1965. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Conf. Applications of Computing Methods to Reactor Problems*, Vol. 557.

Zihan Yu, Lifan Wu, Zhiqian Zhou, and Shuang Zhao. 2024. A Differential Monte Carlo Solver For the Poisson Equation. In *Proc. SIGGRAPH (Conference track)*.

Yonghao Yue, Kei Iwasaki, Bing-Yu Chen, Yoshinori Dobashi, and Tomoyuki Nishita. 2010. Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 29, 6 (Dec. 2010).

Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. 2021. Monte Carlo Estimators for Differential Light Transport. *ACM Trans. Graph. (Proc. SIGGRAPH)* 40, 4 (Aug. 2021).

Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4 (2020), 143:1–143:19.

Tao Zhang, Xiang Tian, XueChao Liu, JianAn Ye, Feng Fu, XueTao Shi, RuiGang Liu, and CanHua Xu. 2022. Advances of deep learning in electrical impedance tomography image reconstruction. *Frontiers in Bioengineering and Biotechnology* 10 (Dec. 2022).

Shuang Zhao, Frédo Durand, and Changxi Zheng. 2018. Inverse Diffusion Curves Using Shape Optimization. *IEEE Trans. Vis. and Comp. Graph.* 24, 7 (2018), 2153–2166.