Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients

ROHAN SAWHNEY∗, Carnegie Mellon University, USA
DARIO SEYB∗, Dartmouth College, USA
WOJCIECH JAROSZ†, Dartmouth College, USA
KEENAN CRANE†, Carnegie Mellon University, USA

Partial differential equations (PDEs) with spatially-varying coefficients arise throughout science and engineering, modeling rich heterogeneous material behavior. Yet conventional PDE solvers struggle with the immense complexity found in nature, since they must first discretize the problem—leading to spatial aliasing, and global meshing/sampling that is costly and error-prone. We describe a method that approximates neither the domain geometry, the problem data, nor the solution space, providing the exact solution (in expectation) even for problems with extremely detailed geometry and intricate coefficients. Our main contribution is to extend the walk on spheres (WoS) algorithm from constant- to variable-coefficient problems, by drawing on techniques from volumetric rendering. In particular, an approach inspired by null-scattering yields unbiased Monte Carlo estimators for a large class of 2nd-order elliptic PDEs, which share many attractive features with Monte Carlo rendering: no meshing, trivial parallelism, and the ability to evaluate the solution at any point without solving a global system of equations.

Additional Key Words and Phrases: partial differential equations, Monte Carlo methods, volumetric light transport, geometry processing

Reference Format:
Rohan Sawhney∗, Dario Seyb∗, Wojciech Jarosz†, and Keenan Crane†. 2022. Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients. XXX XX, X, Article XX (January 2022), 19 pages.

1 INTRODUCTION
PDEs with spatially-varying coefficients describe a rich variety of phenomena. In thermodynamics, for example, variable coefficients model how heterogeneous composite materials conduct or insulate heat. Much as early algorithms for photorealistic rendering were motivated by predictive lighting design [Ward and Shakespeare 1998], such models can be used to predict and improve thermal efficiency in building design [Zalewski et al. 2010]. Likewise, variable permittivity in electrostatics impacts the design of antennas [Ozdemir 2005] and the simulation of biomolecules [Fahrenberger et al. 2014]; in hydrology, variable transmissivity of water through soil impacts remediation strategies for groundwater pollution [Willmann et al. 2010]. More directly connected to our work, variable coefficients in the light transport equation are used to model heterogeneity in participating media [Novák et al. 2018]. Beyond spatially-varying materials, variable coefficients can also be used to model curved geometry, by using PDE coefficients on a flat domain to encode an alternative Riemannian metric (see Sec. 7.5).

The symbols ∗ and † indicate equal contribution.
Our method computes the exact solution (in expectation) to 2nd-order linear elliptic equations of the form

\[ \nabla \cdot (\alpha(x) \nabla u) + \tilde{\omega}(x) \cdot \nabla u - \sigma(x) u = -f(x) \quad \text{on } \Omega, \]
\[ u = g(x) \quad \text{on } \partial \Omega. \]

Here \( \Omega \) is a region in \( \mathbb{R}^n \), \( \alpha : \Omega \mapsto \mathbb{R}_{\geq 0} \) and \( \tilde{\omega} : \Omega \mapsto \mathbb{R}^n \) are twice-differentiable function and vector field (resp.), and \( \sigma : \Omega \mapsto \mathbb{R}_{\geq 0} \) is continuous (C\(^0\)). These conditions are sufficient to ensure ellipticity. The source term \( f : \Omega \mapsto \mathbb{R} \) and boundary values \( g : \partial \Omega \mapsto \mathbb{R} \) need not be continuous. Fig. 5 illustrates the effect of each term on the solution \( u \); see Sec. 2.2 for further background on PDEs. Note that unlike methods for numerical homogenization [Desbrun et al. 2013], we aim to directly resolve the original, detailed solution (Fig. 2).

Fig. 3. Top: The bottleneck in conventional methods is often not the solve itself, but rather the cost of meshing (here, via Hu et al. [2020]). As in rendering, WoS needs only build a simple bounding volume hierarchy (BVH). Bottom: Conventional methods also sacrifice spatial detail—here destroying key features like blood vessels. 

Fig. 4. So-called “meshless” methods still perform a process akin to global meshing, which can result in spatial aliasing of fine features. One ends up with a mesh-like structure which must satisfy stringent sampling criteria to avoid numerical blowup, and must still solve a large globally-coupled linear system. (Figure adapted from [Pauly et al. 2005, Figure 6].)

Since now the discretization must also be carefully adapted to regions where coefficients exhibit fine detail (Fig. 24). Moreover, many meshing and sampling algorithms for constant-coefficient problems do not work out of the box on spatially-varying problems.

Overall, these challenges make it difficult (if not impossible) to analyze large, heterogeneous systems (as in Fig. 1) which commonly arise in real applications—say, directly analyzing a building information model (BIM) that includes heating ducts, plumbing, insulation, etc., rather than a simplified geometric proxy. Yet scenes of this size and complexity are commonplace in rendering. Why such a big gap between our ability to visualize and analyze complex scenes? A major reason is that rendering has moved away from methods like finite element radiosity [Cohen and Wallace 1993], and toward Monte Carlo methods, both to handle more intricate light transport phenomena—and to avoid difficult meshing problems [Jensen 2001, Chapter 1]. For PDEs however, Monte Carlo techniques have received comparatively little attention.

An attractive alternative are grid-free Monte Carlo methods, which solve PDEs without discretizing the problem domain (Sec. 8.1.3). The starting point is the walk on spheres (WoS) method of Muller [1956], which applies a recursive integral formulation akin to the classic rendering equation [Kajiya 1986]. This approach sidesteps many challenges faced by conventional solvers: it can evaluate the solution at any point without meshing the domain or solving a global system, is trivial to parallelize, and works directly with any boundary representation (implicit surfaces, spline patches, etc.), including low-quality meshes not designed for finite-element analysis. Following Sawhney and Crane [2020], recent work in computer graphics explores how to generalize and accelerate WoS, by drawing inspiration from geometry processing and Monte Carlo rendering [Krayer and Müller 2021; Marschner et al. 2021; Mosberg 2021; Nabizadeh et al. 2021]. To date, however, WoS still handles only a small class of constant-coefficient PDEs, limiting its use in applications.

1.1 Contributions

We generalize WoS to a large set of variable-coefficient PDEs, by establishing a link with recent null-scattering techniques for rendering heterogeneous participating media [Novák et al. 2018]. To our knowledge, there is no other known way to solve such PDEs without spatially discretizing the problem domain (whether by previous WoS methods, or any other means). These methods are appropriate for problems involving the steady-state of a diffusive process (in contrast to, say, the dynamics of large scale deformations). Specifically, we provide:
Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients

- A novel reformulation of variable-coefficient 2nd-order linear elliptic PDEs amenable to Monte Carlo methods.
- Efficient WoS algorithms for these PDEs, inspired by the delta-tracking [Woodcock et al. 1965] and next-flight [Cramer 1978] methods from volume rendering.
- A variance reduction strategy that significantly reduces noise in problems with high-frequency coefficients, based on insights from neutron transport.

In the process, we obtain a precise mathematical picture of the relationship between diffusive-variable-coefficient PDEs and heterogeneous participating media. Since we transform the original problem into a constant-coefficient PDE with a modified source term, we get the usual convergence guarantee for WoS algorithms: the variance of an $N$-sample estimate decreases at a rate $1/N$, with a negligible amount of bias due to the $e$-shell (see [Sawhney and Crane 2020, Section 6.1]). Numerical experiments on several thousand models empirically verify convergence behavior (see Figs. 17, 18, 25 and 29).

Limitations. WoS is still an emerging class of methods that does not yet support all the features of more mature solvers (such as FEM)—for example, it is still not known how to handle general Neumann or Robin boundary conditions (as required for, e.g., linear elasticity). These open problems are largely orthogonal to the questions we address here, and are left to future work. See Sec. 9 for further discussion.

2 BACKGROUND

The derivation of our method depends on concepts from PDE theory, the theory of integral equations, stochastic calculus, and volumetric rendering. Since we expect few readers to be familiar with all of these topics, we provide essential background here. Sec. 3 then describes the basic WoS algorithm, which is the starting point for our variable-coefficient algorithm in Sec. 4. For a gentler introduction to WoS, see Sawhney and Crane [2020, Section 2].

2.1 Notation

For any region $A \subset \mathbb{R}^n$, we use $|A|$ to denote its volume and $\partial A$ to denote its boundary. For any point $x \in A$, $\bar{x}$ denotes the point closest to $x$ on the boundary $\partial A$. Throughout, $\Omega \subset \mathbb{R}^n$ denotes the domain of interest, $B(x)$ is a ball centered on $x$ and contained in $\Omega$, and $\partial \Omega_{\epsilon} := \{ x \in \Omega : |x - \bar{x}| < \epsilon \}$ denotes an $\epsilon$-shell around $\partial \Omega$. We use $\vec{u}$ for a vector field on $\mathbb{R}^n$, and $\nabla$ and $\nabla \cdot$ for the gradient and divergence operators (resp.), so that $\Delta := \nabla \cdot \nabla$ is the negative-semidefinite Laplacian. We use $\mathcal{U}$ to denote the uniform distribution on $[0, 1] \subset \mathbb{R}$, and $\mathcal{N}(x, \sigma)$ for the $n$-dimensional normal distribution with mean $x$ and variance $\sigma$. For brevity, we often omit the arguments of functions (e.g., $f$ rather than $f(x, y)$). We use $\mathbb{E}[X]$ to denote the expected value of any random variable $X$.

2.2 Differential Equations

A partial differential equation (PDE) describes a function $u$ implicitly, via relationships between derivatives in space and time. Since this description is implicit, one must ultimately solve for an explicit function $u$ satisfying this equation—which is the raison d’être for numerical PDE solvers. A standard example is the Laplace equation

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = 0.$$  

Solutions to this equation, called harmonic functions, are very smooth—describing, e.g., the steady-state distribution of heat in a room, or a smooth interpolation of color over an image (see Section 7.4).

Order and Linearity. The order of a PDE is the degree of its highest-order derivatives. E.g., the Laplace equation is 2nd order in space, since it involves 2nd derivatives in $x, y$ and $z$. A PDE is linear if it is a linear polynomial in the function and its derivatives. E.g., the Laplace equation is linear, whereas the inviscid Burger’s equation $\frac{\partial u}{\partial t} - u \frac{\partial u}{\partial x} = 0$ is nonlinear, since $u$ is multiplied by its $x$-derivative.

Boundary conditions. To pin down a specific solution to a PDE, we must typically prescribe boundary conditions. For instance, a Laplace equation with Dirichlet boundary conditions, given by

$$\Delta u = 0 \quad \text{on } \Omega, \quad u = g(x) \quad \text{on } \partial \Omega,$$

describes a smooth function $u$ taking values $g(x)$ at points $x$ along the domain boundary (which describe, e.g., fixed temperature or color). Neumann conditions instead specify derivatives (e.g., flow of heat through the boundary). Note that though our final algorithms handle only Dirichlet conditions, the transformations we develop in Sec. 4 make no assumptions about the type of boundary conditions—and hence could in the future be applied to Neumann problems also.

Source term. Continuing with the heat analogy, a source term $f : \Omega \to \mathbb{R}$ adds additional “background temperature” to a PDE. For instance, a Poisson equation has the form

$$\Delta u = -f(x) \quad \text{on } \Omega,$$

possibly subject to some boundary conditions.

Fig. 5. Effect of each term of Eq. (1) on its solution. Left to Right: a Laplace equation $\Delta u = 0$ smoothly diffuses boundary values $g$ into the interior. A source term $f$ adds interior contributions. The diffusion coefficient $\alpha(x)$ controls the diffusion rate at each point $x$. The transport coefficient $\omega(x)$ causes values to drift along the prescribed directions. Positive/negative values of the screening coefficient $\sigma(x)$ dampen/magnify the solution, respectively.
Absorption. Finally, an absorption term (also known as a screening term) describes a "cooling" of the solution due to the background medium. For instance, a screened Poisson equation is given by
\[ \Delta u - \sigma u = -f(x) \text{ on } \Omega, \] (4)
again subject to boundary conditions.

2.3 Integral Equations

We can revisit the PDEs from the previous section through the lens of recursive integral equations, akin to the classic rendering equation [Kajiya 1986]. The reason for doing so is that (as in rendering) solutions to these equations can be computed via recursive application of Monte Carlo integration, as we’ll discuss in Sec. 3.

Boundary conditions. In particular, the three numbered equations from the previous section are examples of so-called elliptic PDEs—solutions to such PDEs can often be expressed as integrals. An important example is the mean value principle, which says that at each point \( x \), the solution \( u(x) \) to a Laplace equation (Eq. (2)) equals the mean value of \( u \) over any sphere around \( x \):
\[ u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(z) \, dz. \] (5)
Notice that Eq. (5) is a recursive integral: the value of \( u \) at \( x \) depends on (unknown) values at other points \( z \). The "base case" is then effectively provided by the (known) boundary values \( g \).

Source term. For a PDE with a source term \( f \), such as the Poisson equation (Eq. (3)), the integral representation of the solution gains a term
\[ \int_{B(x)} f(y) G(x, y) \, dy, \] (6)
where \( G(x, y) \) is the Green’s function for the PDE. In general, for a linear PDE \( Lu = f \), the Green’s function \( G \) satisfies \( LG = \delta \), where \( \delta \) is a Dirac delta. When \( L \) is the Laplacian \( \Delta \), \( G \) is called the harmonic Green’s function, and describes the result of adding a single “spike” of heat to the source. Importantly, the Green’s function depends on the shape of the domain \( \Omega \), e.g., it will be different for a ball versus all of \( \mathbb{R}^n \). The reason this representation is attractive is that common Green’s functions are often known in closed form—we provide several explicit expressions in Sec. 1 of the supplemental material.

Absorption. Adding an absorption term \( \sigma u \) to our PDE changes only the Green’s function in the integral representation, which we now write as \( G^\sigma \). Fig. 6 shows what this function looks like on a ball for increasing values of the coefficient \( \sigma \), and different points \( x \) inside the ball.

Constant coefficients. Putting all these terms together, the solution to a screened Poisson equation with constant coefficients (Eq. (4)) can be written as
\[ u(x) = \int_{B(c)} f(y) G^\sigma(x, y) \, dy + \int_{\partial B(c)} u(z) P^\sigma(x, z) \, dz. \] (7)
Here we also use the more general off-centered integral formulation of the solution \( u \), where the point of evaluation \( x \) need not coincide with the center of the ball \( B(c) \) [Duffy 2015; Hwang et al. 2015]. In particular, the boundary term now incorporates the Poisson kernel
\[ P^\sigma(x, z). \] Like the Green’s function, the Poisson kernel describes how a "spike" on the boundary affects the solution, and in general is given by the normal derivative of the Green’s function at the boundary (see supplemental material for explicit expressions). For \( x = c \) and \( \sigma = 0 \), \( P^\sigma(x, z) \) reduces to \( \frac{1}{|\partial B(x)|} \), recovering the usual mean value property.

Variable coefficients. Our main variable-coefficient PDE, Eq. (1), will be elliptic as long as the diffusion coefficient \( \alpha(x) \) is strictly positive, and the screening coefficient \( \sigma(x) \) is nonnegative at every point \( x \in \Omega \) [Evans 1998; Friedman and Fu 1975]. (The latter condition is in fact stronger than necessary, but it becomes difficult to check ellipticity when \( \sigma(x) < 0 \).) To date, however, there appears to be no integral representation suitable for computation via WoS. We will develop such a representation in Sec. 4.

2.4 Stochastic Equations

Finally, we can also express the solution to a PDE in terms of the Feynman-Kac formula [Oksendal 2003, Ch. 8], which is a fundamental result in stochastic calculus. Considering this formulation serves two purposes: first, it is more general than (known) integral representations, providing a critical starting point for the new integral formula we develop in Sec. 4. Second, the Feynman-Kac formula can be put in close correspondence with the volume rendering equation (VRE) [Sec. 2.5], providing us with key techniques for numerical integration.

2.4.1 Stochastic Processes. A continuous stochastic process describes the trajectory of a particle taking a continuous “random walk.” A central example is a Brownian motion \( W_t \), characterized by the property that increments \( W_{t+s} - W_t \) follow a normal distribution \( N(0, s) \), and are independent of past values of \( W \) (Fig. 7). More generally, a diffusion process describes the trajectory of a particle moving with velocity \( \bar{v}(x) \), and subjected to random displacements of strength \( \alpha(x) \) (Fig. 8). Any such process solves a stochastic differential equation (SDE) [Higham 2001; Kloeden and Platen 2013] of the form:
\[ dX_t = \bar{v}(X_t) \, dt + \sqrt{\alpha(X_t)} \, dW_t. \] (8)

XXX, Vol. XX, No. X, Article XX. Publication date: January 2022.
The term $dW_t$ represents an infinitesimal Brownian increment that adds Gaussian "noise" to an otherwise deterministic trajectory. More generally, we can imagine the particle has some chance of getting absorbed into the background medium (see Fig. 8, right); a function $\sigma(x)$ is used to describe the strength of absorption at each point. Absorbing random walks are quite similar in spirit to light rays in a purely absorbing homogeneous medium—as we’ll discuss in Sec. 2.5.

### 2.4.2 Feynman-Kac Formula

Notice that the parameters $\sigma, \omega, \sigma$ governing the behavior of a diffusion process $X_t$ are in 1-1 correspondence with the coefficients of our main PDE (Eq. (1)). The Feynman-Kac formula connects these two pictures by expressing the solution to the PDE as an expectation over random trajectories of $X_t$.

**Boundary term.** We can start to build up the relationship between PDEs and stochastic processes by considering just a simple Laplace equation $\Delta u = 0$ with Dirichlet boundary values $g$ (Eq. (2)). In this case, [Kakutani 1944] states that

$$u(x) = \mathbb{E}[g(W_T)],$$

where $T$ is the (random) time where $W_t$ first hits the domain boundary $\partial \Omega$. In other words, the solution to a Laplace equation is just the average value "seen" by random walkers starting at $x$.

**Source term.** For a PDE with a source term, such as the Poisson equation $\Delta u = -f$, the random walker will also pick up a source contribution of the form (see [Øksendal 2003, Ch. 9])

$$\mathbb{E} \left[ \int_0^T f(W_t) \, dt \right].$$

Intuitively, this value captures the average heat “felt” by random walkers starting at $x$, along their overall trajectory.

**Absorption.** To model the effect of absorption, as in a screened Poisson equation $\Delta u - \sigma u = -f$, we can incorporate the coefficient $\sigma$ into the boundary and source terms to get

$$\mathbb{E} \left[ e^{-\sigma T} g(W_T) \right] \quad \text{and} \quad \mathbb{E} \left[ \int_0^T e^{-\sigma t} f(W_t) \, dt \right],$$

respectively [Øksendal 2003, Ch. 8]. Notice that larger values of $\sigma$ yield smaller contributions. For a spatially-varying absorption coefficient, we can simply replace $e^{-\sigma T}$ with $e^{-\int_0^T \sigma(W_t) \, dt}$.

**Feynman-Kac.** Finally, to account for variable diffusion $\sigma(x)$ and drift $\omega(x)$, we can replace the Brownian motion $W_t$ with a general diffusion process $X_t$, à la Eq. (8). Combining the above expressions for boundary, source, and absorption terms we arrive at the full Feynman-Kac formula

$$u(x) = \mathbb{E} \left[ \int_0^T e^{-\int_0^t \sigma(X_s) \, ds} f(X_t) \, dt + e^{-\int_0^T \sigma(X_t) \, dt} g(X_T) \right].$$

### 2.4.3 Relationship Between Stochastic and Integral Equations

The stochastic viewpoint also provides a useful interpretation of the Green’s function $G^\sigma$ and Poisson kernel $P$. Suppose in both cases we restrict the domain to a ball $B(x)$. Then $G^\sigma(x, y)$ describes the (unnormalized) probability density that a random walker starting at $x$ passes through any point $y$ inside the ball; $P^\sigma(x, z)$ describes the probability that the walker exits through any given point $z$ on the boundary $\partial B(x)$ (assuming it isn’t absorbed first inside $B(x)$). For instance, when there is no absorption ($\sigma = 0$), a Brownian motion $W_t$ starting at the ball center $x = c$ will exit through points on the boundary sphere $\partial B(x)$ with uniform probability (by symmetry). In this case, the Poisson kernel also reduces to a constant function, in line with the mean value property (Eq. (5)).

### 2.5 Volume Rendering

In computer graphics, the radiative transport equation (RTE) [Chandrasekhar 1960] is used to describe the behavior of light in heterogeneous media that absorb, scatter and emit radiation (Fig. 9, left). Unlike Eq. (1), the RTE is only 1st-order in space. It states that the radiance $L(x, \vec{\omega})$ in a medium at a point $x$ and along a fixed direction $\vec{\omega}$ is given by:

$$\vec{\omega} \cdot \nabla L - \sigma(x)L = -f(x, \vec{\omega}, L) \quad \text{on } \Omega,$$

$$L = g(x, \vec{\omega}, L) \quad \text{on } \partial \Omega.$$  \tag{13}$$

This equation provides a recursive definition of $L$, since the source term $f(x, \vec{\omega}, L)$ depends on the radiance $L(x, \vec{\omega})$ in-scattered at $x$ (as well as any emission $L_e(x, \vec{\omega})$); likewise, the function $g(x, \vec{\omega}, L)$ describes radiance leaving the boundary. The spatially-varying extinction coefficient $\sigma(x)$ specifies the density of (scattering or absorbing) particles at $x$.

The integral form of the RTE is called the volume rendering equation [Pharr et al. 2016, Ch. 15.1] in computer graphics. It describes the radiance $L(x, \vec{\omega})$ from parameterized points $x_t := x - t\vec{\omega}$ along
a ray of length \( d \) in a heterogeneous medium:

\[
L(x, \vec{o}) = \int_{0}^{d} e^{-\int_{0}^{s} \sigma(x_\ell) \, ds} f(x_\ell, \vec{o}, L) \, ds + e^{-\int_{0}^{d} \sigma(x_\ell) \, ds} g(x_d, \vec{o}, L). \tag{14}
\]

Unlike the Feynman-Kac formula in Eq. (12), the VRE is deterministic. However, one faces the same challenges in estimating the VRE directly with Monte Carlo as with the Feynman-Kac formula. In particular, both naively resolving the transmittance function \( e^{-\int_{0}^{d} \sigma(x_\ell) \, ds} \) and using explicit time-stepping to generate points \( x_t \) along a ray lead to biased results.

**Delta tracking.** The delta tracking method \([\text{Raab et al. 2008; Woodcock et al. 1965}]\) enables unbiased Monte Carlo estimation of the VRE. Mathematically, delta tracking simply shifts the spatially-varying 0th order term \( \sigma(x) L \) in Eq. (13) to the source term on the righthand side \([\text{Galtier et al. 2013; Kutz et al. 2017}]\), while introducing the constant coefficient \( \hat{\sigma} \) on both sides of the equality. This yields

\[
\vec{o} \cdot \nabla L - \partial_t L = - (f(x, \vec{o}, L) + (\hat{\sigma} - \sigma(x)) L), \tag{15}
\]

with a corresponding integral expression:

\[
L(x, \vec{o}) = \int_{0}^{d} e^{-\int_{0}^{s} \hat{\sigma} \, ds} f(x_\ell, \vec{o}, L) \, ds + e^{-\int_{0}^{d} \hat{\sigma} \, ds} g(x_d, \vec{o}, L). \tag{16}
\]

Eqs. (15) and (16) are equivalent to Eqs. (13) and (14), but crucially, Eq. (16) is amenable to Monte Carlo estimation since its transmittance function \( e^{-\hat{\sigma} t} \) does not vary spatially. As shown in Fig. 10, the conceptual idea is to fill a heterogeneous medium with fictitious null matter so that the resulting medium has a constant combined density \( \hat{\sigma} \coloneqq \max(\sigma(x)) \) everywhere. Doing so enables perfect, closed-form importance sampling of the transmittance by drawing samples \( t \) from the probability density \( \sigma e^{-\sigma t} \). When colliding with null matter, probabilistically continuing in the forward direction and downweighting the radiance by \( \hat{\sigma} - \sigma(x) \) (see definition of \( f' \) in Eq. (15)) correctly accounts for spatial variations in \( \sigma(x) \).

In Sec. 4, we derive a generalized mean value expression for the PDE in Eq. (1) by applying the delta tracking transformation to the Feynman-Kac formula. This enables the design of modified WoS algorithms in Sec. 5 inspired by techniques for path tracing heterogeneous media.

### 3 WALK ON SPHERES

In this section describe the basic walk on spheres algorithm. Walk on spheres (WoS) was originally developed by Muller [1956] to solve the Laplace equation (Eq. (2)), but has since been extended to a broader set of PDEs. Sec. 3.2 reviews WoS estimators for constant-coefficient PDEs, which serve as building blocks for the variable-coefficient extension we introduce in Sec. 4.

#### 3.1 Monte Carlo Integration

WoS is a Monte Carlo estimator for the solution to a PDE. In general, a Monte Carlo estimator approximates an integral using random samples of the integrand. In particular, for any \( (L^1) \) integrable function \( \phi : \Omega \to \mathbb{R} \), the quantity

\[
I := \int_{\Omega} \phi(x) \, dx
\]

can be approximated by the sum

\[
\hat{I}_N := \frac{1}{N} \sum_{i=1}^{N} \phi(X_i), \quad X_i \sim p,
\]

where the \( X_i \) are independent random samples drawn from any probability density \( p \) that is nonzero on the support of \( \phi \). In this paper we will express all our estimators as single-sample estimators \( \hat{I} \) (dropping the subscript \( N = 1 \) for brevity), with the expectation that their values will be averaged over many trials to improve accuracy.

Importantly, although \( \hat{I}_N \) is called an "estimator", it does not provide merely an estimate—that, a well-designed estimator may give the exact value of the integral, in expectation. More precisely, \( \hat{I}_N \) is unbiased if \( \mathbb{E}[\hat{I}_N] = I \) for any number of samples \( N \), and consistent if the error \( \hat{I}_N - I \) goes to zero as \( N \to \infty \) with probability one \([\text{Veach 1997, Section 1.4.4}]\). Error is more often quantified by the variance \( \text{Var}[\hat{I}_N] := \mathbb{E}[(\hat{I}_N - \mathbb{E}[\hat{I}_N])^2] \), i.e., the average squared deviation from the expected value. As long as \( \phi \) has finite variance, an unbiased estimator is automatically consistent (by the central limit theorem), with variance going to zero at a rate \( O(1/N) \).

#### 3.2 The Walk on Spheres Algorithm

Suppose we want to evaluate the solution to a basic Laplace equation \( \Delta u = 0 \) with Dirichlet boundary conditions \( g \) (Eq. (2)) at some point \( x_0 \in \Omega \). The mean value formula (Eq. (5)) says that \( u(x_0) \) is equal to the average of \( u \) over any ball \( B(x_0) \subset \Omega \); alternatively, Kakutani’s
principle (Eq. (9)) says that \( u(x_0) \) equals the expected value of \( u \) where trajectories of random walkers first hit the ball boundary:

\[
\hat{u}(x_0) = \frac{1}{|B(x_0)|} \int_{\partial B(x_0)} u(z) \, dz = \mathbb{E}[u(W_t)].
\]

Both perspectives point to the same strategy for estimating \( u(x_0) \): uniformly sample a point \( x_1 \) on a ball around \( x \). If \( x_1 \) is extremely close to the domain boundary (i.e., within the \( \epsilon \)-shell \( \Omega_\epsilon \)), grab the boundary value \( g(x_1) \). Otherwise, evaluate \( u(x_1) \). This reasoning leads to the recursive WoS estimator

\[
\hat{u}(x_k) = \begin{cases} 
  g(x_k), & x_k \in \partial \Omega, \\
  \hat{u}(x_{k+1}), & \text{otherwise}.
\end{cases}
\]

Here the point \( x_{k+1} \) is drawn from a uniform distribution on the largest sphere centered at \( x_k \), helping us reach the boundary in a small number of steps. Fig. 11, left depicts one possible “walk” taken by this algorithm. As shown in Fig. 29 (and in [Sawhney and Crane 2020, Figure 14]) terminating this walk in the \( \epsilon \)-shell introduces a negligible bias, which diminishes at a rate of \( O(1/\log \epsilon) \) [Binder and Braverman 2012].

**Source term.** To incorporate a source term \( f \), we also need to estimate the integral from Eq. (6) over each ball \( B(x_k) \) in the walk, and add this contribution to the estimate for \( u(x_k) \). Here again we use a single-sample estimate at a point \( y_{k+1} \) (Fig. 11, right). Though \( y_{k+1} \) could be sampled uniformly, Sawhney and Crane [2020, Section 4.2] show that better results are obtained by importance sampling the source \( f \), Green’s function \( G \), or combining strategies via multiple importance sampling [Veach and Guibas 1995].

**Absorption.** Finally, to incorporate a constant absorption term \( \sigma u \), as in the constant-coefficient screened Poisson equation (Eq. (4)), we simply need to adopt the corresponding Green’s function and Poisson kernel (given in supplemental material).

Unfortunately, a generalized mean value expression like Eq. (7) is not readily available for PDEs with variable coefficients, making it unclear how to use WoS to solve the PDE in Eq. (1). However, the solution to this PDE can be described by the stochastic Feynman-Kac formula (12). In the next section, we exploit the structural similarities between Feynman-Kac and the volume rendering equation (14) to derive a generalized mean value expression for Eq. (1).

### 4 OUR METHOD

The key insight of our method is that while WoS cannot directly deal with spatially-varying coefficients, it can easily solve PDEs with a spatially-varying source term \( f \). Therefore, to derive a generalized mean value expression for Eq. (1), we first apply a series of transformations (shown in Fig. 12) that convert this PDE into an equivalent constant-coefficient screened Poisson equation. These transformations shift all spatial variability in the coefficients \( \alpha(x) \), \( \sigma(x) \) to the source term on the right-hand side, resulting in a PDE where all the differential operators have constant coefficients (Eq. (23)). From a stochastic integral perspective, this is equivalent to reformulating the Feynman-Kac formula (Eq. (12)) purely in terms of Brownian motion (instead of a generic diffusion process). We then leverage the known mean value formulation of the resulting screened Poisson PDE in Eq. (7) to develop Monte Carlo estimators that can be simulated with WoS.

In this section, we assume for clarity of exposition that the transport coefficient \( \tilde{\sigma}(x) = 0 \) over the entire domain. The structure of the transformations we apply remains unchanged for the generic case, which we provide in Appendix A.3. For readers not interested in the derivation, Eq. (26) gives an integral formulation for the solution to Eq. (1), while Sec. 5 describes two variants of WoS to estimate this integral.

#### 4.1 Transformations

As a first step, we apply the product rule to the 2nd-order operator in Eq. (1). We then divide the resulting equation by \( \alpha(x) \), and apply the identity \( \nabla \ln(\alpha(x)) = \nabla \alpha(x) / \alpha(x) \) to get:

\[
\Delta u + \nabla \ln(\alpha(x)) \cdot \nabla u - \frac{\sigma(x)}{\alpha(x)} u = -\frac{f(x)}{\alpha(x)}. \tag{20}
\]

Notice that unlike the lower order terms in this intermediate equation, the 2nd-order term \( \Delta u \) no longer depends on a spatially-varying coefficient.

**Girsanov transformation.** A Girsanov transformation [Oksendal 2003, Ch. 8] is a powerful tool in stochastic calculus that allows a random process to be re-expressed under a change of probability measure, e.g., from a generic diffusion process \( X_t \) to an ordinary Brownian motion \( W_t \). As shown in Appendix A.1, applying this transformation to Eq. (20) eliminates the 1st order operator entirely, shifting all of its spatial variability into the coefficient in the lowest (0th) order term:

\[
\Delta U - \sigma'(x) U = f'(x) \quad \text{on } \Omega, \tag{21}
\]

\[
U = g'(x) \quad \text{on } \partial \Omega.
\]

XXX, Vol. XX, No. X, Article XX. Publication date: January 2022.
Here,
\[ U := \sqrt{\alpha(x)} u, \quad g'(x) := \sqrt{\alpha(x)} g(x), \quad f'(x) := \sqrt{\alpha(x)} f(x), \]
and \[ \sigma'(x) := \frac{\sigma(x)}{\alpha(x)} + \frac{1}{2} \left( \frac{\Delta \alpha(x)}{\alpha(x)} - \frac{|\nabla \ln(\alpha(x))|^2}{2} \right). \]
Eq. (21) is equivalent to our original PDE with variable coefficients in Eq. (1), which can be verified by substituting the expressions for \( U, g', f' \) and \( \sigma' \) back into this equation. Unlike the Feynman-Kac formula in Eq. (12), the stochastic integral expression for this PDE,
\[
U(x) = \mathbb{E} \left[ \int_0^T e^{-\int_0^t \sigma'(W_s) \, ds} f'(W_t) \, dt + e^{-\int_0^t \sigma'(W_s) \, ds} g'(W_t) \right],
\tag{22}
\]
depends solely on the trajectories of a Brownian random walk \( W_t \). We deal with the spatially-varying screening coefficient \( \sigma'(x) \) next.

**Delta tracking.** At this point, the only remaining term on the left-hand side of Eq. (21) with a variable coefficient is the 0th order term \( \sigma'(x)U \). We now apply the delta tracking transformation (Sec. 2.5) to shift this heterogeneity to the right-hand side, since we know how to express the solution of a PDE with a variable source term in integral form. In doing so, we also introduce an auxiliary coefficient \( \tilde{\sigma} > 0 \) into Eq. (21) by subtracting the term \( \tilde{\sigma}U \) on both sides of the equality. This results in a PDE with the same structure as a screened Poisson equation:
\[
\Delta U(x) - \tilde{\sigma}U(x) = - (f'(x) + (\tilde{\sigma} - \sigma'(x))U(x)) \quad \text{on} \quad \Omega,
\tag{23}
\]
\[
U(x) = g'(x) \quad \text{on} \quad \partial \Omega.
\]
Note that even though only constant coefficients now appear on the left-hand side, no approximation of any kind has been introduced. However, unlike a typical PDE the solution \( U \) still appears on the right-hand side (see the definition of \( f'(x, U) \)). In Sec. 5, we account for this dependence on \( U \) in the recursive definition of our estimators by using a strictly positive value for \( \tilde{\sigma} \).

Like the transformed VRE in Eq. (16) and the stochastic integrals in Eq. (11), the Feynman-Kac expression for Eq. (23),
\[
U(x) = \mathbb{E} \left[ \int_0^T e^{-\tilde{\sigma}t} f'(W_t, U) \, dt + e^{-\tilde{\sigma}t} g'(W_t) \right],
\tag{24}
\]
also has a transmittance function \( e^{-\tilde{\sigma}t} \) that no longer varies spatially.

### 4.2 Generalized Mean Value Formulation

We can now express the solution \( U \) of Eq. (23), at a point \( x \) inside a ball \( B(c) \subset \Omega \) using the integral formulation of the constant coefficient screened Poisson equation from Eq. (7):
\[
U(x) = \int_{B(c)} f'(y, U) G^\sigma(x, y) \, dy + \int_{\partial B(c)} U(z) P^\sigma(x, z) \, dz.
\tag{25}
\]
As a last step, we make the substitution \( U = \sqrt{\alpha(x)} u \) from Eq. (21) to express this integral in terms of the original solution variable \( u \).

This yields:
\[
\begin{align*}
\bar{u}(x_k) & := \frac{1}{\sqrt{\alpha(x_k)}} \left( \int_{B(c)} f'(y, \sqrt{\alpha} u) G^\sigma(x, y) \, dy + \int_{\partial B(c)} U(z) P^\sigma(x, z) \, dz \right),
\end{align*}
\tag{26}
\]
This integral can be estimated with Monte Carlo using WoS, as we show next. Unlike Eq. (7), this formulation is recursively defined in both the volume and boundary integrals over the ball \( B(c) \).

### 4.3 PDE Estimation

A single sample Monte Carlo estimate \( \bar{u} \) for Eq. (26), at a point \( x_k \in B(c) \), is given by:

\[
\begin{align*}
\text{evaluate volume term with prob. } \mathbb{P}^B & \quad \int_{B(c)} f'(y_{k+1}, \sqrt{\alpha} \bar{u}) G^\sigma(x_k, y_{k+1}) \, dy_{k+1} + \int_{\partial B(c)} \bar{u}(x_{k+1}) P^\sigma(x_k, x_{k+1}) \, dz_{k+1}, \\
\text{evaluate boundary term with prob. } \mathbb{P}^B & \quad \int_{\partial B(c)} \bar{u}(x_{k+1}) P^\sigma(x_k, x_{k+1}) \, dz_{k+1}.
\end{align*}
\tag{27}
\]
where \( y_{k+1} \) and \( x_{k+1} \) are points sampled inside and on the surface of the ball \( B(c) \) from the probability densities \( \mathbb{P}^B \) and \( \mathbb{P}^B \) (resp.). \( \mathbb{P}^B \) and \( \mathbb{P}^B \) represent the probabilities with which the volume and boundary integrals are sampled. Given that the solution is recursive in both integrals, naïvely estimating both terms in Eq. (27) (with, e.g., \( \mathbb{P}^B = \mathbb{P}^B \)) will result in an exponentially growing number of spheres to keep track of in every walk, since each sphere branches into two additional spheres (see inset). This continues until one or both of the points \( y_{k+1} \) and \( x_{k+1} \) are contained in the epsilon shell \( \partial \Omega \) around the boundary. In the next section, we avoid branching walks by designing two modified WoS algorithms that vary in their choice for \( \mathbb{P}^B \) and \( \mathbb{P}^B \). In Sec. 2 of the supplemental material, we describe how to also use these algorithms to estimate the spatial gradient of Eq. (26).

### 5 ALGORITHMS

The past few decades of volume rendering research have given rise to a plethora of algorithms for solving the VRE [Novák et al. 2018]. The reason for this development is that algorithms generally optimize different performance metrics to address the heterogeneity in the input coefficients, trading off between the bias, variance and speed with which the solution is estimated. As a result, the effectiveness of any one algorithm often depends on the particular coefficients in a scene. The situation is similar when solving variable-coefficient PDEs like Eq. (1) with Monte Carlo. Like the unidirectional estimator in Georgiev et al. [2019, Eq. 14], we provide a unified integral framework based on Eq. (27) in which to develop different WoS algorithms for diffusive PDEs. In particular, we devise two variants inspired by the delta tracking [Woodcock et al. 1965] and next-flight
5.1 The Delta Tracking Variant of Walk on Spheres

To avoid branching, the delta tracking variant of WoS takes advantage of a special property of the Poisson kernel \( P^B(x_k, x_{k+1}) \) for a screened Poisson equation when \( x_k \) lies at the center of a ball \( B(c) \). In particular, \( P^B(x_k, x_{k+1}) \) equals \( 1 - \hat{\sigma}[G^B(x_k)] \) for positive values of the auxiliary coefficient \( \hat{\sigma} \), with \( \hat{\sigma}[G^B(x_k)] \) taking values in the range \((0, 1)\). The quantity \( |G^B(x)| \) represents the integrated value of the Green’s function over \( y \) on \( B(x) \) (see Sec. 1.1 in the supplemental). One can thus use \( p_B := 1 - \hat{\sigma}[G^B(x_k)] \) as the probability of sampling the boundary term in Eq. (27), and \( p_B := 1 - p'^B \) as the probability of sampling the volume term instead. This choice results in the following non-branching procedure inside the domain \( \Omega \):

\[
\tilde{u}(x_k) := \frac{1}{\hat{\sigma}(x_k)} \left[ \frac{\sqrt{x_k}}{\sqrt{\sigma(x_k)}} \frac{\hat{u}}{\sqrt{G^B(x_k)}} \right] \hat{u}(x_{k+1}) + \mathcal{U} \leq \hat{\sigma}[G^B(x_k)] \text{ otherwise.} \tag{28}
\]

Here we choose the densities \( p_B = G^B(x_k, y_{k+1}) / |G^B(x_k)| \) and \( p'^B = 1 / |dB(x_k)| \) to importance sample \( y_{k+1} \) and \( x_{k+1} \) (resp.). If the point \( x_k \) is contained in the epilon shell of \( x_k \), then the boundary data \( g(x_k) \) is used as the value of the estimate \( \tilde{u}(x_k) \), and the walk is terminated; see Fig. 13.

The auxiliary coefficient \( \hat{\sigma} \) serves as the only free parameter in this algorithm. It can be prescribed any positive value which ensures \( \hat{\sigma} > 0 \) (note that \( \hat{\sigma} = 0 \) if \( \hat{\sigma} = 0 \)). The value it is assigned however affects the variance of the estimator. The delta tracking method for volume rendering bounds the value of the positive extinction coefficient \( \sigma(x) \) in Eq. (14) over the entire medium by setting \( \hat{\sigma} = \max(\sigma(x)) \). Numerically, this choice enables closed-form probabilistic sampling of volumetric “events” (absorption, scattering or null) inside the medium, as well as reflections off the boundary. We choose the parameter \( \hat{\sigma} \) in an analogous manner for the delta tracking variant of WoS by bounding the variable screening coefficient \( \sigma'(x) \) (which shows up inside the definition of \( f' \)) from Eq. (23). Unlike rendering, \( \sigma'(x) \) can take on both positive and negative values inside the domain \( \Omega \) for arbitrary input coefficients \( \sigma(x) \) (and \( \hat{\sigma}(x) \); see Appendix A). We therefore use \( \max(\sigma'(x)) = \min(\sigma'(x)) \) as the default value for \( \hat{\sigma} \), which enables us to use the Green’s function and Poisson kernel for a constant coefficient screened Poisson equation to sample the next random point in the walk; see Fig. 14.

More recent volume rendering research [Georgiev et al. 2019; Novák et al. 2014] treats \( \hat{\sigma} \) as a control variate, rather than a bound, which is chosen based on the profile of the input coefficients. In combination with clever choices for the densities \( p^B \) and \( p'^B \), this gives rise to a broader set of more efficient algorithms compared to delta tracking. We leave such extensions to future work.

5.2 The Next-Flight Variant of Walk on Spheres

The primary weakness of the delta tracking variant of WoS is its deteriorating performance for increasing values of the parameter \( \hat{\sigma} \), see Fig. 16. In particular, the algorithm takes shorter steps per walk as the Green’s function for each ball \( B(x) \) in Eq. (27) becomes more localized with larger \( \hat{\sigma} \) (Fig. 6), and the likelihood of sampling a point inside a ball becomes larger than sampling one on its surface \((p^B > p'^B)\). As a result, the number of distance queries needed to find the largest sphere inside any domain \( \Omega \) shoot up significantly. As with ray intersections for path tracing, distance queries are often the main computational bottleneck forWoS.

To address this issue, we propose another variant of WoS based on the next-flight method [Cramer 1978] in volume rendering. At a high level, this technique always takes large steps inside \( \Omega \) even when the value of \( \hat{\sigma} \) is large. As depicted in Fig. 15, it does so by sampling a point \( x_{k+1} \sim p'^B \) on \( dB(x_k) \) to evaluate the boundary term in Eq. (27) with unit probability (i.e., \( p'^B = 1 \)). To compute \( \tilde{u}(x_k) \), it also evaluates the volume term over \( B(x_k) \) with probability \( p^B = 1 \). However, rather than just evaluating one volume term, we expand out the recursive definition of \( \tilde{u} \) in the volume term repeatedly over \( B(x_k) \) itself. This “unrolling” results in two series expressions \( \tilde{T} \) and \( \tilde{S} \) containing contributions from both the boundary and volume terms (see below). Reusing the same sample point \( x_{k+1} \) across all entries in \( \tilde{T} \) yields a new single point estimate for \( \tilde{u}(x_k) \):

\[
\tilde{u}(x_k) := \frac{1}{\sqrt{\hat{\sigma}(x_k)}} \left( \sqrt{\hat{\sigma}(x_{k+1})} \tilde{T}(x_k, x_{k+1}) + \tilde{S}(x_k) \right), \tag{29}
\]

as the default value for \( \hat{\sigma} \), which enables us to use the Green’s function and Poisson kernel for a constant coefficient screened Poisson equation to sample the next random point in the walk; see Fig. 14.

More recent volume rendering research [Georgiev et al. 2019; Novák et al. 2014] treats \( \hat{\sigma} \) as a control variate, rather than a bound, which is chosen based on the profile of the input coefficients. In combination with clever choices for the densities \( p^B \) and \( p'^B \), this gives rise to a broader set of more efficient algorithms compared to delta tracking. We leave such extensions to future work.

5.2 The Next-Flight Variant of Walk on Spheres

The primary weakness of the delta tracking variant of WoS is its deteriorating performance for increasing values of the parameter \( \hat{\sigma} \); see Fig. 16. In particular, the algorithm takes shorter steps per walk as the Green’s function for each ball \( B(x) \) in Eq. (27) becomes more localized with larger \( \hat{\sigma} \) (Fig. 6), and the likelihood of sampling a point inside a ball becomes larger than sampling one on its surface \((p^B > p'^B)\). As a result, the number of distance queries needed to find the largest sphere inside any domain \( \Omega \) shoot up significantly. As with ray intersections for path tracing, distance queries are often the main computational bottleneck forWoS.

To address this issue, we propose another variant of WoS based on the next-flight method [Cramer 1978] in volume rendering. At a high level, this technique always takes large steps inside \( \Omega \) even when the value of \( \hat{\sigma} \) is large. As depicted in Fig. 15, it does so by sampling a point \( x_{k+1} \sim p'^B \) on \( dB(x_k) \) to evaluate the boundary term in Eq. (27) with unit probability (i.e., \( p'^B = 1 \)). To compute \( \tilde{u}(x_k) \), it also evaluates the volume term over \( B(x_k) \) with probability \( p^B = 1 \). However, rather than just evaluating one volume term, we expand out the recursive definition of \( \tilde{u} \) in the volume term repeatedly over \( B(x_k) \) itself. This “unrolling” results in two series expressions \( \tilde{T} \) and \( \tilde{S} \) containing contributions from both the boundary and volume terms (see below). Reusing the same sample point \( x_{k+1} \) across all entries in \( \tilde{T} \) yields a new single point estimate for \( \tilde{u}(x_k) \):

\[
\tilde{u}(x_k) := \frac{1}{\sqrt{\hat{\sigma}(x_k)}} \left( \sqrt{\hat{\sigma}(x_{k+1})} \tilde{T}(x_k, x_{k+1}) + \tilde{S}(x_k) \right), \tag{29}
\]
As before, the subscript \( \bar{\sigma} \) differentiates the steps in a walk, while the superscript \( j \) indicates the points \( x^j \sim p^B \) sampled inside \( B(x_k) \) to evaluate the series \( \tilde{T}(x^0, z) \) and \( \tilde{S}(x^0) \). The number of terms \( M \) in both series is not predetermined—rather the running “throughput” value \( \prod_{i=0}^{j-1} W(I) \) is used as a Russian Roulette probability to determine \( M \). Branching is avoided by reusing the same boundary estimate \( \tilde{u}(x_{k+1}) \) across all entries in \( \tilde{T} \).

The next-flight variant of WoS does not require any distance queries to evaluate \( \tilde{T} \) and \( \tilde{S} \) inside \( B(x_k) \). This performance benefit does however come at the cost of increased correlation in \( \tilde{T} \) due to the reused value of \( \tilde{u}(x_{k+1}) \); see Fig. 17. In terms of parameters, the choice of how to pick \( \bar{\sigma}, p^B \) and \( p^{BB} \) remains unchanged from Sec. 5.1. Notice though that the Green’s functions \( G^B(x^j, x^{j+1}) \) and Poisson kernels \( p^B(x, z) \) in this algorithm are no longer evaluated at the center of each ball \( B(x_k) \) for \( j > 0 \), but rather at arbitrary points inside \( B(x_k) \). Sec. 1.2 and Sec. 1.4 of the supplemental material provide expressions for evaluating and sampling off-centered versions of these quantities.

Where
\[
\tilde{T}(x^0, z) = \sum_{j=0}^{M} \frac{p^B(x^j, z)}{p^B(z)} \prod_{i=0}^{j-1} W(I),
\]
\[
\tilde{S}(x^0) = \sum_{j=0}^{M} \frac{f(x^j)}{\alpha(x^j)} \prod_{i=0}^{j-1} W(I),
\]
\[
W(I) = G^B(x^j, x^{j+1}) (\bar{\sigma} - \sigma^*(x^{j+1})) p^B(x^{j+1}).
\]

As before, the subscript \( k \) differentiates the steps in a walk, while the superscript \( i \) indicates the points \( x^i \sim p^B \) sampled inside \( B(x_k) \) to evaluate the series \( \tilde{T}(x^0, z) \) and \( \tilde{S}(x^0) \). The number of terms \( M \) in both series is not predetermined—rather the running “throughput” value \( \prod_{i=0}^{j-1} W(I) \) is used as a Russian Roulette probability to determine \( M \). Branching is avoided by reusing the same boundary estimate \( \tilde{u}(x_{k+1}) \) across all entries in \( \tilde{T} \).

The next-flight variant of WoS does not require any distance queries to evaluate \( \tilde{T} \) and \( \tilde{S} \) inside \( B(x_k) \). This performance benefit does however come at the cost of increased correlation in \( \tilde{T} \) due to the reused value of \( \tilde{u}(x_{k+1}) \); see Fig. 17. In terms of parameters, the choice of how to pick \( \bar{\sigma}, p^B \) and \( p^{BB} \) remains unchanged from Sec. 5.1. Notice though that the Green’s functions \( G^B(x^j, x^{j+1}) \) and Poisson kernels \( p^B(x, z) \) in this algorithm are no longer evaluated at the center of each ball \( B(x_k) \) for \( j > 0 \), but rather at arbitrary points inside \( B(x_k) \). Sec. 1.2 and Sec. 1.4 of the supplemental material provide expressions for evaluating and sampling off-centered versions of these quantities.

6 VARIANCE REDUCTION

As shown in Fig. 17, Fig. 18, Fig. 25 and Fig. 29, our WoS algorithms demonstrate the expected Monte Carlo rate of convergence when estimating the solution to variable-coefficient PDEs. Here we present a variance reduction strategy to reduce noise in the PDE estimates.

Weight Window. Our algorithms use the Monte Carlo estimator in Eq. (27) to solve variable-coefficient PDEs. For every step in a walk, the value of the estimate \( \tilde{u}(x_k) \) is scaled by a multiplicative weighting factor (with a value possibly greater than 1). For instance, in the delta tracking variant of WoS, \( \tilde{u} \) is weighted either by \( \left(1 - \frac{\sigma^*(x_k+1)}{\bar{\sigma}}\right) \sqrt{\frac{\partial^2 u(x_k+1)}{\partial \sigma^2}} \) or \( \sqrt{\frac{\partial^2 u(x_k+1)}{\partial \sigma^2}} \) based on whether the volume or the boundary term is sampled (resp.). These weights serve as a source of variance in the estimator; see Fig. 18. This is especially true for large values of the bounding parameter \( \bar{\sigma} \), since a high average number of steps per walk can result in a cumulative weight that is either very small or very large. To address this issue, we recommend using a weight window, a commonly employed variance reduction tool in neutron transport [Hoogenboom and Légrády 2005], to keep the weighting factor roughly constant for any walk.

A weight window achieves variance reduction through a combination of Russian Roulette and splitting (see inset). The former terminates walks with small weights, since it is often not worthwhile completing walks whose likely contribution to the estimator is small. The latter splits walks with large weights into multiple new equally weighted walks. This prevents any single walk from having a large weight, while encouraging better exploration of the domain through the newly spawned walks. We adopt a simple version of the weight window strategy that uses a static window size \([w^-, w^+]\), where \(w^-\) and \(w^+\) are the minimum and maximum cumulative weight (resp.) any walk is allowed to have. In our implementation, we use \([0.5, 1.5]\). A walk whose cumulative weighting factor \( w \) at any step lies within

![Fig. 17. As with volume rendering, the effectiveness of our WoS algorithms varies based on the profile of the input coefficients. Here we show solution estimates computed with the delta tracking and next-flight variants on two reference analytic problems of the form \( \Delta u - \sigma(x) u = f(x) \) with the same solution but different absorption coefficients \( \sigma(x) \) and \( \sigma(x) + 10 \) from Fig. 16. Top Row: The delta tracking estimator demonstrates lower variance than the next-flight estimator due to higher sample correlation in the latter. Bottom Row: Large values of \( \sigma(x) \) (and hence \( \bar{\sigma} \)) cause the efficiency of the delta tracking estimator to drop, since more distance queries are needed to achieve the same variance as the next-flight estimator.](image-url)
the window is allowed to continue without modification. A Russian Roulette survival probability of \( w/w^- \) determines whether to terminate the walk when \( w < w^- \). Otherwise, the walk is split into \( m := w/w^+ \) new walks, each with a weight of \( w/m \) when \( w > w^+ \). Since \( m \) is generally not an integer, we use the expected value splits approach of [Booth 1985] where \( n := \lfloor m \rfloor \) walks are chosen with probability \( n + 1 - m \), and \( n + 1 \) walks are chosen otherwise.

Fig. 18 highlights the effectiveness of a static weight window in reducing variance in problems with high frequency coefficients and large bounding parameters \( \tilde{\sigma} \). In the neutron transport and rendering literature, specifying the window size adaptively has been shown to provide improvements in computational efficiency by up to an order of magnitude over static windows [Booth and Hendricks 1984; Vorba and Krivánek 2016; Wagner and Haghighat 1998]. We leave this optimization to future work.

7 IMPLEMENTATION AND RESULTS

In this section, we discuss practical considerations pertaining to the modified WoS algorithms we present. We also highlight the unique benefits of our approach on a range of example problems inspired by engineering and design applications.

7.1 Algorithm Inputs

Walk on spheres requires computing the distance to the boundary of a domain to find the largest empty sphere in each step of a random walk. Distance queries can be accelerated via a spatial hierarchy such as a bounding volume hierarchy (BVH) or an octree [Ericson 2004]. These acceleration structures can be built quickly and without much memory across a large variety of geometric representations of the domain boundary, e.g., polygonal meshes, NURBS/subdivision surfaces, implicit surfaces and constructive solid geometry [Museth 2013; The Embree developers 2013]. Our CPU implementation currently uses a basic axis-aligned BVH without vectorization, though recent GPUs provide opportunities for further acceleration [Burgess 2020].

Our WoS algorithms for variable-coefficient PDEs additionally require computing the gradient and Laplacian of the diffusion and transport coefficients \( \alpha(x) \) and \( \tilde{\alpha}(x) \) (see Sec. 4.1 and Appendix A.3). These derivatives can be computed using any standard technique (e.g., auto-differentiation). Like most volume rendering algorithms [Novák et al. 2018], we also require computing a bounding parameter \( \tilde{\sigma} := \max(\sigma'(x)) - \min(\sigma'(x)) \) over the domain.

7.2 Geometric Flexibility and Iterative Feedback

Monte Carlo methods are popular in rendering because they provide instantaneous feedback and easily work with a wide variety of geometric representations. This enables engineers, scientists and artists to quickly iterate on their designs, however complex they may be. The same is true for the solvers we present in this paper.

We implemented a GPU version of our solver in the Unity game engine, allowing a user to work with variable-coefficient PDEs interactively (Fig. 19), even on scenes of immense complexity (Fig. 1). We represent the domain geometry using signed distance fields and can use this same representation to interactively edit the scene geometry, render it using sphere tracing [Hart 1996], and solve a PDE using our WoS estimators—all without resorting to any meshing.

Figure 1 shows a collection of objects scattered procedurally to construct an infinitely large yet aperiodic scene. Meshing this entire scene is not possible since it requires infinite storage. In fact, even meshing a subset of the scene is prohibitively expensive (see Fig. 24). Since our method operates directly on the signed distance field and supports local point evaluation, we can compute the solution in only the region visible to the camera and resolve high frequency coefficients despite the large scale of the scene.

In Fig. 19 we show how a user can interactively edit the scene geometry via constructive solid geometry operations while seeing the solution to the variable coefficient PDE on a slice plane. We
perform a single random walk for each point on the slice plane each time we render a frame. When the geometry or coefficients change, we discard prior samples, otherwise we accumulate samples and average. Our solver quickly produces a converged solution, since it runs at 60 frames per second, meaning that it can produce 60 samples per point per second. As a result, users get instant feedback when they change problem inputs, enabling them to guide the design process effectively without any interruptions.

7.3 Integration with other Monte Carlo solvers

One benefit of Monte Carlo solvers is that point evaluation makes it trivial to combine solvers to estimate the solution to nested or coupled problems. As an example of this, we show how to use the Monte Carlo estimators presented in this paper to accelerate volumetric rendering.

The VRE we presented in Sec. 2.5 is already solved almost exclusively using Monte Carlo methods [Novák et al. 2018], at least in the context of offline rendering. Unfortunately, this is expensive for dense, high-albedo media like milk or marble, where light paths can undergo thousands of scattering events. In these cases, it is common to use a diffusion equation like Eq. (1) to model light transport, because it is much faster [Jensen et al. 2001]. The resulting 3D heterogeneous diffusion problem can be computed using classical deterministic solvers [Arbree et al. 2011; Koerner et al. 2014], but these methods have seen limited use in rendering due to their need for meshing, global solves, and great care in coupling the diffusion solver with the overarching Monte Carlo rendering algorithm. Practical approaches have typically relied on approximations like homogeneity and local planarity via the dipole and its variants [d’Eon and Irving 2011; Donner et al. 2008; Habel et al. 2013; Jensen et al. 2001]—which typically introduce more error than the diffusion assumption itself. Our approach instead allows us to couple a Monte Carlo VRE solver with a fully heterogeneous 3D diffusion solver while avoiding many of these compromises.

Our hybrid algorithm uses a simple heuristic to switch on the fly between solving the RTE via classical volumetric path tracing (VPT) and solving a diffusion equation using our WoS estimators. At each path vertex inside the medium, we generate a free-flight distance (as in VPT) and compute the distance to the medium boundary (as in WoS). We then continue the path with VPT or WoS based on which distance is larger. A path can freely change back and forth between these two modes, but in practice the algorithm will prefer the more accurate VPT steps near the boundary, while large WoS steps allow the algorithm to make rapid progress through the interior of the medium. This is conceptually very similar to shell tracing [Moon et al. 2008; Müller et al. 2016] in rendering and condensed history neutron transport [Fleck and Canfield 1984]: each WoS step essentially aggregates an arbitrary number of VPT steps.

In Fig. 20 we compare this hybrid method to VPT on a heterogeneous marble bust, showing that even this simple proof-of-concept heuristic can achieve significant error reduction at equal medium lookups. Note that unlike techniques based on the dipole approximation, our diffusion solver accounts for the 3D heterogeneity inside the medium.

7.4 Heterogeneous Diffusion Curves

WoS is attractive for diffusion curves [Orzan et al. 2008], since it provides real-time progressive previews that are easily implemented on the GPU [Quilez 2020], can be applied to a zoom-in without first computing a coarse global solution (as done by Orzan et al. [2008, Section 3.2.4]), and avoids aliasing of fine features due to grid resolution [Sawhney and Crane 2020, Figure 16]. Our method makes it possible to generalize classic diffusion curves by also painting a source \( f(x) \), diffusion coefficient \( \alpha(x) \), and absorption coefficient \( \sigma(x) \)—Fig. 21 shows one example. The added benefit is that, unlike constant-coefficient diffusion images, the source term \( f(x) \) is no longer severely blurred—enabling one to add interesting decals or background texture, while still smoothly diffusing color over regions without such details. Variable absorption \( \sigma(x) \) helps to further emphasize detail, since the strength of the source contribution is roughly \( 1/\sigma(x) \). Akin to "texture shaders" [Bowers et al. 2011; Prévost et al. 2015], this enables an enriched design space spanning
diffusion curves and traditional 2D graphics, though these prior approaches achieved this via alpha blending.

7.5 Walk on Curved Surfaces
A basic hypothesis of the original WoS algorithm is that a random walk exits every point on the boundary of a ball with equal probability (Sec. 3.2). However, on surfaces with non-constant curvature this hypothesis no longer holds: intuitively, more walkers will escape through the "valleys" than through the "mountains." As a result, standard WoS cannot be used for many algorithms in geometric and scientific computing that need to solve equations on a surface/shell.

Our variable-coefficient scheme enables WoS to be applied to curved surfaces for the first time. In particular, consider any surface expressed as a conformal parameterization \( f : \mathbb{R}^2 \supset U \rightarrow \mathbb{R}^3; \) conformal means that \( f \) distorts the surface by a uniform scaling \( \lambda(x) \) at each point \( x \in U \), i.e., \( J_f^T J_f = \lambda(x) I \), where \( I \) is the identity and \( J_f \) is the Jacobian of \( f \). The Laplacian \( \Delta_f \) of the curved surface is then related to the ordinary Laplacian via \( \Delta = \lambda \Delta_f \). Hence, we can solve PDEs on the curved surface by replacing the usual diffusion coefficient \( a(x) \) with \( \lambda(x) \lambda(x) \). Fig. 22 shows several examples; for periodic domains (like the torus) our walks simply “wrap around.” In theory, every surface admits a conformal parameterization (by the uniformization theorem [Abikoff 1981]), but in practice many important surfaces used in engineering (such as NURBS or other spline patches) are expressed in non-conformal coordinates. To directly handle such patches, we would need to extend our method to anisotropic diffusion coefficients—an important topic for future work. Note also that earlier work on diffusion curves for surfaces uses free-space 2D Green’s functions [Sun et al. 2012], which in general provide only a rough proxy for the true Green’s functions of the curved surface.

8 RELATED WORK AND COMPARISONS
8.1 PDE Solvers
There are innumerable methods for solving variable-coefficient PDEs—here we examine core trade-offs between methods, highlighting unique benefits of Monte Carlo that complement existing solvers. Note that some methods homogenize coefficients to a coarser model that approximates fine-scale behavior [Abdulle et al. 2012; Durlofsky 1991; Dykaar and Kitanidis 1992; Efendiev and Hou 2009; March et al. 2021]. We instead seek to directly resolve fine details—in fact, Monte Carlo methods like ours may help to accurately compute parameters for homogenization.

8.1.1 Finite Element Methods (FEM). All finite element methods (including meshless and boundary element methods) adopt a common mathematical framework: given a linear PDE \( L u = f \), find the best approximation to \( u \) within a finite-dimensional function space. E.g., standard Galerkin FEM considers an approximation \( \tilde{u} := \sum_{i=1}^n u_i \phi_i \) in some basis \( \phi_1, \ldots, \phi_n : \Omega \rightarrow \mathbb{R} \), where the \( u_i \in \mathbb{R} \) are unknown coefficients. Letting \( (u, v) := \int_\Omega u(x)v(x) \, dx \) denote the \( L^2 \) inner product, one then seeks a \( \tilde{u} \) satisfying
\[
(\tilde{u}, \phi_j) = (f, \phi_j), \quad j = 1, \ldots, n,
\]
i.e., such that \( \tilde{u} \) looks exactly like the true solution \( u \) when restricted to the subspace \( V = \text{span}(\{\phi_i\}) \) (but may have error in directions orthogonal to \( V \)). To solve this equation, we can rewrite Eq. (30) as
\[
\sum_{i=1}^n u_i (L \phi_i, \phi_j) = \sum_{i=1}^n f_i (\phi_i, \phi_j).
\]
The inner products between basis functions and their derivatives define elements of the mass and stiffness matrices (resp.), and must typically be approximated via numerical quadrature.

From this perspective, the only difference between flavors of FEM is the choice of basis functions \( \phi_j \). Hence, all finite element methods come with a common set of challenges:
- They must all solve a globally-coupled system of equations.
- They are all prone to spatial aliasing in the geometry, solution, boundary conditions, source terms, and/or coefficients, since any finite basis \{\phi_j\} provides limited spatial resolution.
- They all demand spatial discretization (meshing or sampling) to define bases \( \phi_i \), which can be costly and error prone (Fig. 3).

XXX, Vol. XX, No. X, Article XX. Publication date: January 2022.
Fig. 23. With both FEM and MFEM, local aliasing of high-frequency boundary data yields large global errors in the solution, demanding significant refinement. In contrast, WoS always captures the global solution (even with very few samples); error instead manifests as high-frequency noise.

In contrast, WoS can directly evaluate the solution at any point without meshing/sampling, and without performing a global solve. Moreover, it does not suffer from aliasing in the solution or problem data, since it need not restrict functions to a finite-dimensional subspace (see especially Fig. 23).

**Mesh-Based FEM.** Most often, FEM bases $\phi_i$ are defined via polyhedral mesh elements. Quickly and robustly meshing large, detailed and/or imperfect geometry (e.g., with self-intersections) is an ongoing “grand challenge,” where even state-of-the-art methods can struggle (Fig. 3). This problem gets harder if the mesh must also be refined for spatially-varying coefficients: even with intelligent adaptive mesh refinement (AMR) [Zienkiewicz and Zhu 1992a,b], meshing quickly becomes prohibitive (Fig. 24). More recent a priori $p$-refinement does not help, since it considers only element quality and not spatial frequencies in the solution or problem data [Schneider et al. 2018]. WoS bypasses the meshing grand challenge entirely, needing only a BVH (for closest point queries), which uses minimal memory and can be built in a fraction of a second—even for totally degenerate geometry (see [Sawhney and Crane 2020, Figure 2]).

**Meshless FEM.** Though meshless FEM (MFEM) may sound like a natural alternative to grid-free Monte Carlo, the term “meshless” is a bit of a misnomer: MFEM does not need a polyhedral mesh, but must still discretize the domain by carefully arranging nodes on the domain interior (Fig. 4). Nodes are then associated with bases $\phi_i$, such as a radial basis functions (RBFs), to build mass and stiffness matrices. To couple bases with overlapping support one must identify neighbors—forming a global graph structure similar in size and complexity to a polyhedral mesh. Node locations must satisfy criteria that are often just as difficult and delicate to enforce as mesh quality criteria [Li and Liu 2007, Ch. 3], and often require global optimization akin to mesh smoothing [Slak and Kosec 2019].

Moreover, just as one bad element can ruin an FEM solution, bad node placement can lead to catastrophic failure (e.g., NaNs in the solution—see Fig. 25, top left). To mitigate spatial aliasing one can adaptively sample nodes—but unlike mesh-based FEM, adaptive refinement for MFEM is poorly understood (lacking, e.g., rigorous convergence guarantees). Finally, typical MFEM bases are approximating rather than interpolating, complicating the enforcement of boundary conditions [Fries et al. 2004; Nguyen et al. 2008]; some MFEM methods hence modulate bases by a distance-like function [Shapiro and Tsukanov 1999], but still effectively spatially discretize functions on the interior by choosing a finite basis $\{\phi_i\}$.

As shown in Fig. 25, a more serious challenge with meshless FEM is stagnation: until only very recently [Bayona et al. 2019, 2017; Flyer et al. 2016], MFEM methods might fail to converge without careful problem-specific tuning of parameters such as neighborhood size. More damning is the fact that increasing the neighborhood size does not always make the solution better (see Fig. 26). Moreover, whereas convergence of adaptive FEM is rigorously understood [Mekchay and Nochetto 2005], there is a dearth of corresponding results for adaptive MFEM schemes—especially important for problems with fine details in geometry and/or coefficient functions. In practice, MFEM also requires far denser mass/stiffness matrices than those used in mesh-based FEM, while often providing less-accurate results. For instance, methods such as RBF-FD with polynomial augmentation [Flyer et al. 2016] that converge under refinement require bases of order at least 2. On the whole, MFEM is not known for its reliability—in stark contrast, walk on spheres guarantees that the expected solution is equal to the true solution of the smooth PDE, without any parameter tuning whatsoever. Moreover, unlike MFEM, WoS is truly “meshless” at no point in the algorithm does one require a global sampling or meshing of the domain.

Finally, though meshless FEM has been around for a long time, it has not seen nearly as much use in practice as mesh-based methods.
Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients • XX:15

Fig. 25. Convergence criteria for meshless FEM methods are not well-understood. In practice, methods often fail to converge under refinement (shown here for the same variable-coefficient problem on watertight models generated from the Thingi10k dataset [Hu et al. 2020; Zhou and Jacobson 2016]) and/or show extremely large variation in error. Both mesh-based FEM and Monte Carlo methods come with mathematically rigorous convergence guarantees and behave much more predictably under refinement.

Fig. 26. In practice, it can be difficult to find reliable parameters for meshless FEM—for instance, increasing neighborhood size often increases error in an unpredictable way. In contrast, WoS requires no parameter tuning.

Fig. 27. Meshless FEM must solve a much denser linear system than even standard FEM—whereas WoS avoids solving a global system altogether.

(e.g., with very few open source or commercial packages available relative to mesh-based FEM).

Boundary Element Methods (BEM). Boundary element methods approximate the solution using bases functions \( \phi_i \) associated only with elements of a boundary mesh (such as free-space Green’s functions). These methods draw a natural comparison with walk on spheres, since they need not discretize the interior of the domain. However, there is a significant difference in capabilities: whereas WoS easily handles problems with source terms and spatially-varying coefficients on the domain interior, basic BEM ignores these terms altogether (see Fig. 28). In order to handle general interior terms, one must couple BEM with a second, interior solver such as FEM, MFEM, or FD—incorporating all the same challenges [Coleman et al. 1991; Costabel 1987; Partridge et al. 2012]. Moreover, even for problems involving only boundary terms, BEM must discretize the boundary geometry, leading to spatial aliasing in both boundary data and geometry. Unlike FEM/MFEM, BEM must solve a globally-coupled dense system of equations, demanding special techniques like hierarchical matrix approximation [Hackbusch 2015] to obtain reasonable performance.

8.1.2 Finite Difference (FD) Methods. The main conceptual difference that distinguishes all finite difference methods from finite element methods is that the degrees of freedom are no longer coefficients in a finite basis—instead, they represent point samples of the unknown function at nodal points. Derivatives are then approximated essentially via Taylor series approximation: if one imagines that there is a function interpolating the nodal values, then the derivatives are given in terms of standard finite difference formulas.

On the one hand, FD schemes are attractive due to the simplicity of implementation on a regular grid. They are however less ideal for PDEs with nonuniform coefficients, where the uniformity of grid cells can lead to significant numerical diffusion [Umansky et al. 2005], spurious negative values [Sharma and Hammett 2007], and locking/stagnation [Babuška and Suri 1992]. Another major challenge is spatial adaptivity: for many PDEs, using a fine grid over the whole domain is overkill. In order to adaptively refine solutions (e.g., due to the impact of variable coefficients) one can switch to a hierarchical structure like an octree [Gibou et al. 2018; Losasso et al. 2006]. Octrees mitigate some of the performance advantage of working with a regular grid, and significantly increase the complexity of implementation. FD methods also struggle with accurate handling of boundary conditions due to the fact that cell boundaries are now axis-aligned [Causon and Mingham 2010].

On the whole, finite differences suffer from the same major challenges as finite element methods: one must spatially discretize the domain, the boundary conditions, the source term, and the coefficients of the operator, invariably leading to either aliasing or oversampling. Moreover, one must solve a globally-coupled system of
equations over the entire domain, rather than concentrating computational effort only at points of interest (as in Monte Carlo).

Material Point Methods. Material point methods [Jiang et al. 2016], such as PIC [Harlow and Welch 1965], FLIP [Brackbill and Ruppel 1986; Zhu and Bridson 2005], APIC [Jiang et al. 2015], and MPM [Sulsky et al. 1995] are popular for time-dependent computational mechanics problems involving large scale deformation (fluids, plasticity, etc.). These methods are also sometimes referred to as “meshless”, but they are not (in general) MFEM schemes as defined in Sec. 8.1.1. Rather, these methods use particles to approximate advection, and a background grid to solve elliptic problems (such as pressure projection in fluids). Critically, for the problems we consider here (time-independent elliptic PDEs), there is no advection component, and MPM reduces to simply solving elliptic equations on a grid—with the same tradeoffs as discussed above.

8.1.3 Stochastic Methods. Not all PDE solvers need to discretize space—the notable exception are Monte Carlo methods based on continuous random processes such as Brownian motion. The stochastic approach to deterministic boundary value problems, discussed in Sec. 2, centers on the simulation of random walks that in aggregate solve a large class of 2nd-order elliptic PDEs [Øksendal 2003]. This formulation enables local point evaluation of the PDE solution and has found extensive use in scientific disciplines such as mathematical finance [Black and Scholes 1973; Cox et al. 1985; Merton 1971; Merton and Samuelson 1992], computational physics and chemistry [Gillespie 1977; Grebenkov 2007; Mascagni and Simonov 2004a,b] and optimal control [Kalman 1960; Kappen 2007] (albeit often on simple geometric domains).

Discretized Random Walks. In place of WoS, one might try approximating Feynman-Kac by directly simulating the diffusion process \( X_t \) with explicit time stepping [Higham 2001; Kloeden and Platen 2013], akin to ray marching [Tuy and Tuy 1984]:

\[
X_{k+1} = X_k + \alpha (X_k) h + \sqrt{\sigma(X_k)} (W_{k+1} - W_k). \tag{31}
\]

However, this approach introduces several sources of error. E.g., walks often leave the domain and must be clamped to the boundary: shrinking \( h \) reduces error, but significantly slows down computation (Fig. 29). Also, nonlinear functions \( \phi \) do not in general commute with expectation \((\mathbb{E}[\phi(X)]) \neq \phi(\mathbb{E}[X])\), hence it is not clear how to estimate the function \( \exp(-\int_0^t \sigma(X_s) \, ds) \) in Eq. (12) in an unbiased way. Bias is exacerbated in problems with variable diffusion and transport coefficients, which implicitly modify the ideal step size. In contrast, the \( \varepsilon \)-shell in WoS incurs only a miniscule bias at the very end of the walk, leading to far less error overall (29, top); the size of \( \varepsilon \) also has very little effect on performance (29, bottom). Note that as with ray marching in rendering [Kettunen et al. 2021], discretized walks do still exhibit fairly predictable and low variance, as long as some bias is tolerable.

Continuous Random Walks. A variety of so-called grid-free Monte Carlo methods have been developed for simulating random walks without spatial discretization. The chief example is the WoS method described in Sec. 3; variants of this idea include walk on rectangles [Deaconu and Lejay 2006] and walk on boundaries [Sabelfeld and Simonov 2013]. Zagajac [1995] describes an alternative grid-free scheme based on shooting rays, albeit only for the basic Laplace equation \( \Delta u = 0 \). WoS has been applied in a limited capacity to problems with piecewise constant coefficients [Lejay and Maire 2013; Maire and Nguyen 2016]. Ours is the first grid-free method for PDEs with continuously-varying diffusion, transport, and screening coefficients.

8.2 Volume rendering

While Monte Carlo solvers have been fairly niche in the PDE community, they are the method of choice for solving the radiative transfer equation in both neutron transport [Spanier and Gelbard 1969] and more recently rendering [Novák et al. 2018; Pharr et al. 2016]. Of particular interest to us are the so-called null-scattering methods, the most simple being delta (or Woodcock)-tracking [Coleman 1968; Raab et al. 2008; Woodcock et al. 1965], where a heterogeneous medium is filled with fictitious “null-scattering particles”, which transform it into a mathematically equivalent problem where the total density is constant. Early null-scattering methods were physically motivated, but hard to analyze in a mathematical framework. Recent work [Galtier et al. 2013; Georgiev et al. 2019; Kutz et al. 2017; Miller et al. 2019] has derived null-scattering versions of the RTE that allow for the construction of new estimators within the same framework, but without the need for a clear physical interpretation. These estimators can be used to exploit known medium properties, for example by introducing control variates [Kutz et al. 2017; Novák et al. 2014]. More recently, this framework has been built upon further [Jonsson et al. 2020; Kettunen et al. 2021] and looks to offer a solid foundation for future volumetric rendering work.
expressing our estimators in Sec. 5 in the same way, we believe that future work in volumetric rendering will be easily transferable to our Monte Carlo PDE algorithms as well.

9 LIMITATIONS AND FUTURE WORK

Our method is not without limitations. Much like volume rendering, variance can increase with increasing values of the parameter $\sigma$. Unlike rendering however, the large magnitude of $\sigma(x)$ in Eq. (21) (and hence $\sigma$ if it is bounding) generally results from derivatives of the transport and diffusion coefficients. In applications involving sharp changes in these coefficients, the derivatives essentially serve as edge detection filters. However, there is plenty of inspiration to take from volume rendering to address this limitation, in particular by bounding the spatially-varying coefficients locally, and by porting other lower variance estimators for the VRE [Georgiev et al. 2019; Novák et al. 2018] to solve diffusive PDEs. Furthermore, Monte Carlo techniques not commonly used in rendering but employed in SDE literature such as Multi-Level Monte Carlo [Giles 2015] should help accelerate run-time performance. Adaptive weight windows likely provide significant variance reduction [Booth and Hendricks 1984; Vorba and Krivánek 2016; Wagner and Haghhighat 1998].

Finally, many PDEs, whether constant or variable coefficient, involve Neumann boundary conditions, which remain to be integrated into the walk on spheres framework.

Moving forward, we believe insights from this paper provide building blocks for WoS estimators for an even larger set of PDEs, e.g., with anisotropic coefficients [Boscaini et al. 2016] or certain nonlinear PDEs. Ultimately, we are optimistic our estimators will open the door to new applications where engineers and scientists can worry less about discretization, and directly analyze models of nearly any complexity—just as in rendering today.

REFERENCES

Assyr Abdulle, E. Weinan, Björn Enquist, and Eric Vanden-Eijnden. 2012. The heterogeneous multiscale method. *Acta Numerica* 21 (2012), 1–87.

William Asikoff. 1981. The Uniformization Theorem. *The American Mathematical Monthly* 88. 8 (1981).

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

Adam Arleve, Bruce Walter, and Kavita Bala. 2011. Heterogeneous Subsurface Scattering Using the Finite Element Method. *IEEE TVCG* 17, 7 (July 2011), 956–969.

Ivo Babuška and Manil Suri. 1992. On locking and robustness in the finite element method. *SIAM J. Numer. Anal.* 29, 5 (1992), 1261–1293.

Daniel T Gillespie. 1977. Exact stochastic simulation of coupled chemical reactions. *The Journal of chemical physics* 65, 2 (1976), 90–100.

David Boscaini, Jonathan Masci, Emanuele Rodolà, Michael M Bronstein, and Daniel Cremers. 2016. Anisotropic diffusion descriptors. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library. 431–441.

John C. Bowers, Jonathan Leahy, and Rui Wang. 2011. A Ray Tracing Approach to Diffusion Curves. *Proc. EGSR* 30, 4 (2011), 1345–1352.
A.1 The Girsanov Transformation

In stochastic calculus, the Girsanov transformation describes how to change the dynamics of a random process (e.g., a diffusion process) under a change of probability measure. Consider a process \( X_t \) driven by the SDE \( dX_t = \tilde{\sigma}(X_t) \, dt + dW_t \). Then, the Feynman-Kac formula

\[
0 = \left[ \int_0^\tau \mathbb{E} \left[ e^{-\int_0^t \tilde{\sigma}(X_s) \, ds} f(X_t) \right] \right] \left. \frac{d}{dt} X_t \right|_{t=0} = \int_0^\tau \mathbb{E} \left[ e^{-\int_0^t \tilde{\sigma}(X_s) \, ds} f(X_t) \right] \tilde{\sigma}(X_t) \, dt + \mathbb{E} \left[ f(X_\tau) \right] \]  

expresses the solution to the following PDE as a conditional expectation over random trajectories of \( X_t \) inside \( \Omega \):

\[
\frac{1}{2} \Delta u + \tilde{\sigma}(x) \cdot \nabla u - \sigma(x) u = -f(x) \quad \text{on } \Omega, \\
u = g(x) \quad \text{on } \partial \Omega.
\]

The Girsanov transformation makes it possible to change the transport coefficient \( \tilde{\sigma}(x) \) of the random process \( X_t \) by introducing an importance sampling weight \( Z \) into the Feynman-Kac formula in Eq. (32) to keep its solution unchanged. In particular, to simulate \( X_t \) as a Brownian random walk \( W_t \), this transformation provides an equivalent Feynman-Kac formula:

\[
u(x) = \mathbb{E} \left[ \int_0^\tau e^{-\int_0^t \tilde{\sigma}(W_s) \, ds} Z(W_t) f(W_t) \, dt + e^{-\int_0^\tau \tilde{\sigma}(W_t) \, dt} Z(W_\tau) g(W_\tau) \right|_{W_0 = x} \]  

where the importance weight \( Z \) is explicitly given by the expression:

\[
Z(W_t) := e^{\int_0^t \tilde{\sigma}(W_s) \, dw_s - \frac{1}{2} \int_0^t d\hat{\sigma}(W_s)^2 \, ds}
\]

The term \( \int_0^\tau \tilde{\sigma}(W_s) \cdot dw_s \) is called a stochastic integral since it is defined with respect to variations of a Brownian random process \( W_t \). We refer the reader to Øksendal [2003] for a formal definition of this integral and its properties. We describe how to evaluate this integral next.

A.2 The Chain Rule of Stochastic Calculus

The stochastic calculus counterpart of the chain rule, known as Itô’s lemma, serves as a useful tool for evaluating stochastic integrals. Given a twice differential function \( \gamma(x) : \mathbb{R}^n \to \mathbb{R} \), Itô’s lemma states that its differential \( dy \), as a function of a Brownian process \( W_t \), is given by:

\[
dy(W_t) = \nabla \gamma(W_t) \cdot dw_t + \frac{1}{2} \sum_i \gamma''(W_t) \, dw_i.
\]

Integrating this expression over time and rearranging the terms, we get:

\[
\int_0^\tau \nabla \gamma(W_t) \cdot dw_t = \gamma(W_\tau) - \gamma(W_0) - \frac{1}{2} \sum_i \gamma''(W_t) \, dw_i.
\]

This integrated version of Itô’s lemma allows us to re-express the importance weight \( Z \) without any stochastic integral. In particular, if the transport coefficient \( \tilde{\sigma}(x) \) takes the form of a gradient of a scalar field \( \nabla \gamma(x) \), then an alternative expression for \( Z \) is given by:

\[
Z(W_t) = \exp \left( \gamma(W_t) - \gamma(W_0) - \frac{1}{2} \int_0^\tau \left( \Delta \gamma(W_t) + |\nabla \gamma(W_t)|^2 \right) \, dw \right).
\]

A.3 Derivation of Eq. (21)

With this new expression for \( Z \), the modified Feynman-Kac formula in Eq. (34) takes the form:

\[
u(x) = e^{-\gamma(x)} \mathbb{E} \left[ \int_0^\tau e^{-\int_0^t \tilde{\sigma}(W_s) \, ds} f(W_t) \, dt + e^{-\int_0^\tau \tilde{\sigma}(W_t) \, dt} g(W_\tau) \right|_{W_0 = x} \]  

where

\[
f'(x) := e^{\gamma(x)} f(x), \quad g'(x) := e^{\gamma(x)} g(x), \quad \sigma'(x) := \sigma(x) + \frac{1}{2} \left( \Delta \gamma(x) + |\nabla \gamma(x)|^2 \right)
\]

Setting \( U := e^{\gamma(x)} u \), the PDE corresponding to this Feynman-Kac formula is given by:

\[
\frac{1}{2} \Delta U - \sigma'(x) U = -f'(x) \quad \text{on } \Omega, \\
U = g'(x) \quad \text{on } \partial \Omega,
\]

which does not contain a 1st order transport term. In Appendix A.1, we assume \( \sigma(x) := 1 \) since it is always possible to absorb a scalar valued diffusion coefficient into the source function \( f(x) \) and screening coefficient \( \sigma(x) \), as shown with the PDE in Eq. (20). Pattern matching further with Eq. (20), notice that we recover the PDE in Eq. (21) by setting the scalar field \( \gamma(x) \) to \( \frac{1}{2} \ln(\sigma(x)) \).

Finally, we point out that the Girsanov transformation does not only apply when the transport coefficient \( \tilde{\sigma}(x) \) equals just the gradient of a single scalar field. For instance, to use this transformation, \( \tilde{\sigma}(x) \) in Eq. (1) can be defined additively as the gradient of several scalar fields, i.e., \( \sum_i \nabla \gamma_i(x) \).
1 GREEN’S FUNCTIONS AND POISSON KERNELS

Here we provide the Green’s function $G(\sigma; x, y)$ and Poisson kernel $P(\sigma; x, y)$ for a constant coefficient screened Poisson equation on a ball $B(c)$ in 2D and 3D. These quantities are needed to estimate the integral expression in Eq. 26 we derive for variable coefficient PDEs in the paper. Expressions for $\nabla_x G(\sigma; x, y)$ and $\nabla_x P(\sigma; x, y)$ are provided as well to estimate the spatial derivative of Eq. 26. We also describe how to draw samples $y$ inside $B(c)$ from a probability density $p(\sigma)$ that is proportional to the Green’s function. Derivations of $G_{2D}$ and $G_{3D}$ can be found in Duffy [2015].

1.1 Centered Expressions

Assume that the point $x$ lies at the center of a ball $B(c)$ with radius $R$, and let $r := |y - x|$. Then the Green’s function on $B(x)$ in two and three dimensions is given by:

$$G_{2D}(x, y) = \frac{1}{2\pi} \left( K_0(r\sqrt{\sigma}) - \frac{K_0(R\sqrt{\sigma})}{I_0(R\sqrt{\sigma})} I_0(r\sqrt{\sigma}) \right),$$

$$G_{3D}(x, y) = \frac{1}{4\pi} \left( e^{-r\sqrt{\sigma}} \left( \frac{K_1(r\sqrt{\sigma})}{I_2(\sqrt{\sigma})} - \frac{K_1(R\sqrt{\sigma})}{I_2(R\sqrt{\sigma})} \right) \right),$$

where $I_n$, $I_{n+1}$, and $K_n$, $K_{n+1}$ (for $n = 0, 1, 2, ...$) denote modified Bessel functions of the first and second kind (resp.). Routines to efficiently evaluate these functions are available in numerical libraries such as Boost [Schäling 2014] and SciPy [Virtanen et al. 2019].

To compute the probability density $p(R(x, y) := G(\sigma; x, y)/|G(\sigma; x)|)$ associated with these Green’s functions, we need to evaluate the integrated value of $G(\sigma)$ over all $y$ on $B(x)$:

$$|G_{2D}(x)| := \int_{B(x)} G_{2D}(x, y) dy = \frac{1}{\pi} \left( 1 - \frac{1}{I_0(R\sqrt{\sigma})} \right),$$

$$|G_{3D}(x)| := \int_{B(x)} G_{3D}(x, y) dy = \frac{1}{\pi} \left( 1 - \frac{R\sqrt{\sigma}}{\sinh(R\sqrt{\sigma})} \right).$$

The Poisson kernel is defined as the normal derivative of the Green’s function along the boundary, i.e., for any point $z$ on $\partial B(z)$, $P(\sigma; x, z) := \nabla_x G(\sigma; x, z) \cdot \hat{n}(z)$. In two and three dimensions it is given by:

$$P_{2D}(x, z) = \frac{1}{2\pi R} \left( \frac{1}{I_0(R\sqrt{\sigma})} \right),$$

$$P_{3D}(x, z) = \frac{1}{4\pi R^2} \left( \frac{R\sqrt{\sigma}}{\sinh(R\sqrt{\sigma})} \right).$$

Notice that in both dimensions, the Poisson kernel equals $\frac{1-\sigma}{|\partial B(x)|}$.

We exploit this property of the Poisson kernel to develop the delta tracking variant of WoS described in Sec. 5.1 of the paper.

1.2 Off-centered Expressions

The next-flight variant of WoS from Sec. 5.2 in the paper requires off-centered versions of the Green’s function and Poisson kernel. In particular, assume $x$ is an arbitrary point inside $B(c)$, and let $r_- := \min(|x - c|, |y - c|)$ and $r_+ := \max(|x - c|, |y - c|)$. Furthermore, let $\theta$ define the angle between the vectors $x - c$ and $y - c$ in 2D or 3D. Then the off-centered Green’s function on $B(c)$ is given by the infinite series:

$$G_{2D}(x, y) = \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} \cos(n\theta) I_n(r_- \sqrt{\sigma}),$$

$$G_{3D}(x, y) = \frac{1}{4\pi} \sum_{n=0}^{\infty} (2n + 1) P_n(\cos(\theta)) \left( \frac{\pi}{2r_- \sqrt{\sigma}} I_{n+\frac{1}{2}}(r_- \sqrt{\sigma}) \right)^2,$$

where $P_n$ denotes the recursively defined Legendre polynomials. As usual, the Poisson kernel can be computed by evaluating $\nabla_x G(\sigma; x, y) \cdot \hat{n}(z)$ on $\partial B(c)$. We recover the expressions for $G(\sigma)$ and $P(\sigma)$ in Sec. 1.1 when $x$ coincides with the ball center $c$.

In practice, we observe that 100 to 200 terms are required to accurately approximate these series. To avoid this computational burden, we provide approximations for these off-centered quantities. In particular, let $\tilde{u} := x - c$, $\tilde{v} := y - c$ and $\tilde{w} := y - x$. Then in two and three dimensions we have:

$$G_{2D}(x, y) \approx \frac{1}{\pi} \left( Q_{2D}(\tilde{w}) - Q_{2D}(\frac{R^2 - \tilde{u} \cdot \tilde{v}}{R}) \right),$$

$$G_{3D}(x, y) \approx \frac{1}{4\pi} \left( Q_{3D}(\tilde{w}) - Q_{3D}(\frac{R^2 - \tilde{u} \cdot \tilde{v}}{R}) \right).$$

$$P_{2D}(x, y) \approx \frac{1}{2\pi} \left( \frac{2\sqrt{R^2 - \tilde{u} \cdot \tilde{v}}}{|\tilde{w}|} + \frac{1}{2} V_{2D}(\frac{R^2 - \tilde{u} \cdot \tilde{v}}{|\tilde{w}|}) \right),$$

$$P_{3D}(x, y) \approx \frac{1}{4\pi} \left( \frac{2\sqrt{R^2 - \tilde{u} \cdot \tilde{v}}}{|\tilde{w}|} + \frac{1}{2} V_{3D}(\frac{R^2 - \tilde{u} \cdot \tilde{v}}{|\tilde{w}|}) \right),$$

where

$$Q_{2D}(x, y) = \frac{1}{2\pi} \left( \frac{1}{I_0(R\sqrt{\sigma})} \right),$$

$$Q_{3D}(x, y) = \frac{1}{4\pi} \left( \frac{R\sqrt{\sigma}}{\sinh(R\sqrt{\sigma})} \right).$$

XXX, Vol. 1, No. 1, Article. Publication date: February 2022.
These expressions for $G^\sigma$ and $P^\sigma$ are exact when $x$ lies at the center of $B(c)$, but begin to diverge slightly from the true values as $x$ is moved closer to $\partial B(c)$ and the value of coefficient $\sigma$ is decreased; see Fig. 1. In our experiments, we observe that these approximate expressions provide sufficiently accurate results with the next-flight variant of WoS with far less compute, especially when the value of $\sigma$ is decreased.

1.3 Gradient Expressions

In Sec. 2 of this document, we provide an integral expression for the gradient of a PDE solution at a point $x$. To estimate the gradient at the center of $B(x)$, we need to evaluate the gradients of the Green’s function and Poisson kernel. In two and three dimensions they are given by:

$$
\nabla_x G^\sigma_{2D}(x, y) = \frac{(y - x) \sqrt{\sigma}}{2\pi r} \left( K_1(r \sqrt{\sigma}) - \frac{K_0(R \sqrt{\sigma})}{I_0(R \sqrt{\sigma})} I_1(r \sqrt{\sigma}) \right),
$$

$$
\nabla_x G^\sigma_{3D}(x, y) = \frac{(y - x) \sqrt{\sigma}}{4\pi^2} \left( e^{-r \sqrt{\sigma}} \left( 1 + \frac{1}{r \sqrt{\sigma}} \right) - \frac{\cosh(r \sqrt{\sigma}) - \sinh(r \sqrt{\sigma})}{r \sqrt{\sigma}} \right),
$$

$$
\nabla_x P^\sigma_{2D}(x, z) = \frac{(z - x) \sigma}{2\pi R} \left( \frac{1}{R \sqrt{\sigma}} I_1(R \sqrt{\sigma}) \right),
$$

$$
\nabla_x P^\sigma_{3D}(x, z) = \frac{(z - x) \sigma}{4\pi R^2} \left( \frac{1}{R \sqrt{\sigma}} \cosh(R \sqrt{\sigma}) - \frac{\sinh(R \sqrt{\sigma})}{R \sqrt{\sigma}} \right).
$$

1.4 Sampling

To sample from the probability density $p^B \equiv \sigma(x, y) / |G^\sigma(x)|$ associated with the centered Green’s functions $G^\sigma$ in Sec. 1.1, we first pick a direction $\vec{y}$ uniformly on the unit sphere [Arvo 2001]. A radius $r$ is then sampled from the distribution $2\pi r p^B$ in 2D, or $4\pi r^2 p^B$ in 3D, using rejection sampling. The extra factor in front of $p^B$ accounts for the change of measure between polar and Cartesian coordinates. For rejection sampling, we bound the radial density by the following case dependent function:

$$
h(R, \sigma) = \begin{cases} 
\max(2.2 * \max(1/R, 1/\sigma), 0.6 * \max(\sqrt{R}, \sqrt{\sigma})) & R \leq \sigma, \\
\max(2.2 * \min(1/R, 1/\sigma), 0.6 * \min(\sqrt{R}, \sqrt{\sigma})) & \text{otherwise}.
\end{cases}
$$

The final sample point is given by $y = r \vec{y} + x$.

Generating samples from an off-centered Green’s function $G^\sigma(x, y)$ in Sec. 1.2 is more challenging since we do not know of a closed-form expression for $|G^\sigma(x)|$. While using a uniform density $\frac{1}{|B(c)|}$ for $p^B$ suffices for unbiased sampling, more sophisticated techniques to generate samples according to the profile of $G^\sigma$ exist. We recommend the weighted reservoir version of resampled importance sampling provided in [Bitterli et al. 2020, Alg. 3].
ALGORITHM 2: NextFlightEstimate(x)

Input: A point \( x \in \Omega \).
Output: A single sample estimate \( \hat{u}(x) \) of the solution to Eq. 1.
/* Compute distance and closest point to \( x \) on \( \partial \Omega \) */
\( d, \bar{x} \leftarrow \text{DistanceToBoundary}(x) \);
/* Return boundary value \( g \) at \( \bar{x} \) if \( x \in \partial \Omega_c \) */
if \( d < \epsilon \) then return \( g(\bar{x}) \);
/* Initialize series expressions from Eq. 29 */
\( \hat{z} \leftarrow 0, \quad \hat{S} \leftarrow 0; \)
/* Initialize path throughput */
\( W \leftarrow 1; \)
/* Sample random exit point \( z \in \partial B(x) \) used across all entries in \( \hat{T} \) */
\( z \sim \frac{1}{\Omega(x)}; \)
/* Initialize temporary variable to track current sample point inside \( B(x) \) */
\( x_c \leftarrow x \)
while True do
/* Accumulate boundary contribution */
\( \hat{T} + A \frac{p^B(x_c, z)}{p^B(\bar{x}, z)} W; \)
/* Use path throughput as Russian Roulette probability to terminate loop */
\( P_{\text{RR}} = \min(1, W); \)
if \( P_{\text{RR}} < \mu \) then break;
\( W \leftarrow P_{\text{RR}}; \)
/* Sample next random point \( x_n \in B(x) \) */
\( x_n \sim \frac{1}{\Omega(x)} \); 
/* Update path throughput */
\( W \leftarrow \frac{\sigma(x_n)}{\bar{\Omega}(x_n)} \frac{\alpha(x_n)}{\alpha(x)} W; \)
/* Accumulate source contribution */
\( S + \frac{1}{\Omega(x)} \frac{f(x_n)}{\sigma(x_n)} \); 
/* Update current sample point inside \( B(x) \) */
\( x_c \leftarrow x_n \)
end
/* Estimate solution at \( z \in \partial B(x) \) */
return \( \frac{1}{\Omega(x)} \sqrt{\alpha(z)} \hat{T} \text{NextFlightEstimate}(z) + \hat{S}; \)

Sec. 5 in the paper. Furthermore, the parameters \( \bar{\sigma}, p^B, p^{\partial B}, \bar{P}^B \) and \( \bar{p}^{\partial B} \) remain unchanged with either algorithm.

3 PSEUDO-CODE

Here we provide pseudo-code for the two variants of walk on spheres presented in Sec. 5 of the paper. To maintain consistency with the paper, we assume the transport coefficient \( \bar{\sigma}(x) = 0 \) over the entire domain.

ALGORITHM 1: DeltaTrackingEstimate(x)

Input: A point \( x \in \Omega \).
Output: A single sample estimate \( \hat{u}(x) \) of the solution to Eq. 1.
/* Compute distance and closest point to \( x \) on \( \partial \Omega \) */
\( d, \bar{x} \leftarrow \text{DistanceToBoundary}(x); \)
/* Return boundary value \( g \) at \( \bar{x} \) if \( x \in \partial \Omega_c \) */
if \( d < \epsilon \) then return \( g(\bar{x}) \);
/* Estimate source contribution at random point */
\( y \in B(x) \)
\( y \sim \frac{G(x, y)}{G(x)} ; \)
\( \hat{S} \leftarrow \frac{\alpha(x)}{\alpha(x)} f(y) \); 
/* Decide whether to sample volume or boundary term in */
Eq. 27
if \( \mu \sim U \leq \theta |G(x)| \) then
/* Estimate solution at \( y \in B(x) \); adjust estimate by */
null-event contribution from Eq. 21
\( \frac{\alpha(x)}{\alpha(x)} \left( 1 - \sigma'(y) \right) \) DeltaTrackingEstimate(y) + \( \hat{S}; \)
else
/* Estimate solution at random point \( z \in \partial B(x) \) */
\( z \sim \frac{1}{\Omega(x)} \); 
return \( \frac{1}{\Omega(x)} \) DeltaTrackingEstimate(z) + \( \hat{S}; \)
end

REFERENCES

J. Arvo. 2001. Stratified sampling of 2-manifolds. SIGGRAPH Course Notes 29, 2 (2001).
Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 39, 4 (July 2020).
Dean G. Duffy. 2015. Green’s functions with applications. Chapman and Hall/CRC.
B. Schilling. 2014. The boost C++ libraries. XML Press.
P. Virtanen, R. Gommers, and Contributors. 2019. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. arXiv e-prints, Article arXiv:1907.10121 (Jul 2019), arXiv:1907.10121 pages. arXiv:1907.10121