Monte Carlo Geometry Processing: Analysis of a Grid-Free Approach

Project Report - Master MVA

Baran Celik
MVA
Paris, France
baran.celik@ens-paris-saclay.fr

Swann Cordier
MVA
Paris, France
swann.cordier@ens-paris-saclay.fr

Antoine Le Maguet
MVA
Paris, France
antoine.lemaguet@free.fr

Abstract

Conventional Finite Element Methods (FEM) for solving Partial Differential Equations (PDEs) face severe scalability bottlenecks when applied to complex, non-manifold, or dirty geometries due to the necessity of volumetric meshing. This report reviews the *Walk on Spheres* (WoS) algorithm, a grid-free Monte Carlo method that solves linear elliptic PDEs by simulating stochastic diffusion processes. We investigate the theoretical foundations connecting the Feynman-Kac formula to the Mean Value Property and implement the solver for Laplace, Poisson, and variable-coefficient problems. Our experimental results demonstrate the method's unique advantages, including infinite resolution zooming, robustness to "polygon soup" geometry, and ease of parallelization. However, we also highlight critical limitations through stress tests, specifically the "blindness" of random walkers in narrow tunnel geometries and variance explosion in gradient estimation near boundaries. We conclude that while WoS offers a powerful alternative for geometry processing, it currently requires specific acceleration structures to compete with FEM in topologically constrained domains. All results presented in this report were generated using our own Python implementation of the WoS algorithm.

1 Introduction

Second-order elliptic Partial Differential Equations (PDEs), such as the Laplace and Poisson equations, form the mathematical backbone of numerous physical simulations and geometry processing tasks. From simulating heat transfer and electrostatics to smoothing surfaces and computing shape deformations, the ability to solve these equations efficiently is paramount in scientific computing.

For decades, the standard approach to these boundary value problems has been dominated by deterministic, grid-based methods, most notably the Finite Element Method (FEM). While highly effective, FEM relies fundamentally on the spatial discretization of the domain. Generating a high-quality volumetric mesh for modern geometric models, often characterized by non-manifold features, self-intersections, or "polygon soup"—is a computationally expensive and fragile bottleneck. In many workflows, the meshing phase consumes more resources than the numerical solution itself.

This report reviews a grid-free alternative: the **Walk on Spheres (WoS)** algorithm. Originally introduced by **Muller [2]** and recently adapted for computer graphics by **Sawhney and Crane [4]**, this method leverages the stochastic connection between harmonic functions and Brownian motion. By using Monte Carlo integration, the solution can be evaluated at arbitrary points without constructing a global mesh or solving a coupled linear system.

The report is structured to guide the reader from theory to critical analysis. We begin by establishing the mathematical preliminaries and detailing the *Walk on Spheres* algorithm, including its extensions for the Poisson equation and variable-coefficient PDEs using variance reduction techniques. Subsequently, we present our experimental results, demonstrating the method's robustness on complex benchmarks such as "polygon soups" and parametric curves. We conclude with a critical discussion comparing WoS to FEM, highlighting both its unique advantages in local evaluation and its specific limitations regarding topological constraints and gradient estimation.

2 Context and Preliminaries

2.1 Notation and Problem Statement

Throughout this report, we consider a volumetric domain $\Omega \subset \mathbb{R}^n$ with boundary $\partial\Omega$. We focus on linear second-order elliptic PDEs. The prototypical example is the **Poisson equation**:

$$\begin{cases} \Delta u(x) = f(x) & \text{for } x \in \Omega \\ u(x) = g(x) & \text{for } x \in \partial\Omega \end{cases} \tag{1}$$

Where:

- $\Delta = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2}$ is the Laplace operator.
- $f : \Omega \to \mathbb{R}$ is the source term. In case of Laplace equation, we have $f = 0$.
- $g : \partial\Omega \to \mathbb{R}$ prescribes Dirichlet boundary conditions.
- $u : \Omega \to \mathbb{R}$ is the unknown solution we wish to compute.

We also denote $B(x)$ as the ball centered at x with radius R, contained within Ω. The distance to the closest point on the boundary is denoted by $d(x) = \min_{y \in \partial\Omega} |x - y|$. We will write $|A|$ for the volume of a set. More generally, we have used the paper's notation.

2.2 The Conventional Bottleneck

Conventional approaches to PDE-based geometry processing rely fundamentally on the spatial discretization of the domain. Although these methods are mature, they encounter severe scalability and robustness issues when applied to complex geometric models typical of modern applications.

The primary bottleneck lies in the generation of a volumetric mesh. This pre-processing step is computationally expensive and notoriously fragile when dealing with "dirty" geometry containing holes, self-intersections or non-manifold features. For intricate domains, the cost of meshing often dominates the total runtime. For instance, meshing a complex biological structure can require

over 14 hours of computation and 30GB of memory, rendering the approach impractical for rapid iteration.

Furthermore, the requirement for high-quality elements often necessitates geometric simplification. Robust meshing algorithms can smooth out or remove fine-scale details to ensure convergence, leading to discretization errors and loss of critical features. Additionally, FEM requires solving a global linear system that couples all degrees of freedom, which contrasts with the memory-efficient Monte Carlo methods.

2.3 Monte Carlo Estimators

The core insight of the paper is that linear elliptic PDEs can be solved via stochastic calculus. Instead of discretizing space, we utilize the Feynman-Kac formula connections (**Kakutani's Principle** [1]). This principle states that for a harmonic function, the value at a point x is the expected value of the boundary condition g at the location where a Brownian motion starting at x first exits the domain. Mathematically, this relationship is grounded in the **Mean Value Property**. For any ball $B(x) \subset \Omega$ centered at x, the value $u(x)$ is the average of u over the sphere boundary $\partial B(x)$:

$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y)dy. \tag{2}$$

To evaluate the integral in Equation 2, we apply a **Monte Carlo estimator**. If f is the integrand and we draw N independent random samples X_i uniformly on the sphere, the integral is approximated by:

$$F_N = \frac{1}{N} \sum_{i=1}^{N} f(X_i), \quad X_i \sim \mathcal{U}(\partial B(x)). \tag{3}$$

This estimator exhibits three fundamental statistical properties that distinguish it from grid-based approximations:

- **Unbiasedness:** $\mathbb{E}[F_N]$ is equal to the true integral value for any N.
- **Consistency:** The estimator is consistent; as N approaches infinity, the error converges to zero with probability one.
- **Predictable error:** The error is due purely to variance and decreases at a rate of $O(1/\sqrt{N})$. Notably, this convergence rate is independent of the spatial dimension n.

2.4 Closest Point Queries

The fundamental geometric operation in the WoS method is computing the radius of the largest inscribed empty sphere centered at the current walker position x :

$$d(x, \partial\Omega) = \min_{y \in \partial\Omega} \|x - y\|.$$

The way this distance is computed depends on the boundary representation. For a single primitive A, the closest point is given either by the orthogonal projection of x onto A, or by the nearest endpoint if the projection falls outside the primitive. We can adapt it to operation between subset:

$$d(x, A \cup B) = \min(d(x, A), d(x, B)).$$

$$d(x, A \cap B) = max(d(x, A), d(x, B))$$

This property enables geometric flexibility, allowing solvers to operate directly on noisy, non-manifold, or heterogeneous boundaries without requiring interior meshing or topological repair.

For highly detailed boundaries composed of n primitives, computing the minimum distance by naïvely checking all elements scales as $O(n)$, which is prohibitive. Practical solvers therefore rely on spatial acceleration structures, most commonly a **Bounding Volume Hierarchy (BVH)**, which recursively groups primitives into bounding boxes. During a query, subtrees whose bounding volumes lie farther than the best known current candidate can be pruned. This reduces the amortized cost of each closest-point query to approximately: Cost $\approx O(\log n)$, a crucial improvement since Monte Carlo–WoS solvers execute millions of such boundary queries during simulation.

3 The Walk on Spheres Algorithm

3.1 From Brownian Motion to Spheres

While the connection between the Laplace equation and Brownian motion is theoretically sound (Kakutani's Principle), directly simulating a diffusion process via small time-steps Δt is computationally intractable. As $\Delta t \to 0$, the number of steps required to exit the domain approaches infinity.

To overcome this, Muller proposed the **Walk on Spheres (WoS)** algorithm. The core idea exploits the isotropy of Brownian motion and the Mean Value Property. If a random walker is currently at position $x_k \in \Omega$, the walker can construct the largest empty sphere centered at x_k and distance $d(x_k)$. Due to symmetry, the probability of the walker exiting this sphere is uniform over its surface. Consequently, rather than taking many small steps, the walker can "jump" immediately to a point x_{k+1} uniformly sampled from the sphere's surface $\partial B(x_k)$.

This procedure yields a recursive estimator for the solution u at the starting point x_0. If the walk consists of points $x_0, x_1, \ldots, x_N$ where x_N is on the boundary, the estimator for the Laplace equation is simply the boundary value at the exit point:

$$\hat{u}(x_0) = g(x_N) \tag{4}$$

By averaging this result over many independent walks, we converge to the true solution $u(x_0)$.

3.2 Algorithm and Termination

The WoS algorithm proceeds by simulating independent random walks. For a single walk starting at x_0, the walker jumps from sphere to sphere until reaching the boundary.

However, strictly speaking, a walk might never exactly hit the boundary, exhibiting Zeno's paradox as the spheres become infinitesimally small near $\partial\Omega$. To ensure termination, the algorithm employs an **ϵ-shell truncation**: the walk stops when the walker comes within a small tolerance ϵ of the boundary.

Crucially, a single walk provides only a noisy, high-variance estimate of the solution. To obtain a converged solution $u(x_0)$, we must compute the Monte Carlo estimator by averaging the results of N independent walks. The full solver algorithm is detailed below:

Algorithm 1 Monte Carlo WoS Solver (Laplace)

Require: Query point x_0, Boundary $\partial\Omega$, Tolerance ϵ, Number of walks N

$u_{sum} \leftarrow 0$
for $i \leftarrow 1$ **to** N **do**
  $x \leftarrow x_0$ ▷ Start a new walk
  while $d(x, \partial\Omega) > \epsilon$ **do**
    $R \leftarrow d(x, \partial\Omega)$ ▷ Closest Point Query
    $\vec{v} \leftarrow$ UniformSampleUnitSphere()
    $x \leftarrow x + R \cdot \vec{v}$ ▷ Jump to sphere surface
  end while
  $x_b \leftarrow$ ClosestPoint$(x, \partial\Omega)$
  $u_{sum} \leftarrow u_{sum} + g(x_b)$ ▷ Accumulate boundary value
end for
return u_{sum}/N ▷ Return average

3.3 Complexity and Bias

The use of the ϵ-shell introduces a small amount of bias, as the solution is technically evaluated at a distance ϵ from the true boundary. However, the paper notes that this bias is negligible for sufficiently small ϵ (e.g., 10^{-4} to 10^{-6}).

Crucially, the number of steps required to reach this ϵ-shell grows logarithmically with the inverse of the tolerance, $O(\log(1/\epsilon))$. This implies that increasing precision (reducing ϵ) is computationally inexpensive, distinguishing WoS from grid-based methods where refining resolution incurs a polynomial cost increase.

4 Solving the Poisson Equation

We now extend our framework to the Poisson equation $\Delta u = f$, where physical phenomena involve a source term f. In this setting, the solution depends not only on the boundary values but also on the trajectory of the random walk through the domain.

4.1 Stochastic and Integral Representations

From a stochastic perspective, the solution to the Poisson equation can be represented using the Feynman-Kac formula. It corresponds to the expected value of the boundary condition minus the accumulation of source term along the path of a Brownian motion W_t:

$$u(x) = \mathbb{E}_x \left[g(W_\tau) - \int_0^\tau f(W_t) dt \right] \tag{5}$$

where τ is the exit time. Intuitively, the integral term tracks the distribution of where random walks "spend time" inside the domain before exiting. The Green's function $G(x, y)$ effectively acts as a density function for this residence time within a specific ball $B(x)$.

To make this computationally tractable using WoS, we utilize the recursive integral relation on $B(x)$:

$$u(x) = \underbrace{\frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy}_{\text{Boundary Average}} + \underbrace{\int_{B(x)} f(y) G(x, y) dy}_{\text{Source Contribution}} \tag{6}$$

Here, the first term handles the harmonic part (jumping to the surface), while the second term captures the source contribution within the ball volume.

4.2 Monte Carlo Estimator

We cannot evaluate the volume integral exactly during the walk. Instead, we use a **one-sample Monte Carlo estimator**. For a single step at x_k, we sample a point y_k uniformly from the volume of the ball $B(x_k)$. The estimator for the source integral is:

$$F_{vol} \approx |B(x_k)| \cdot f(y_k) \cdot G(x_k, y_k), \quad y_k \sim \mathcal{U}(B(x_k)) \tag{7}$$

where $|B(x_k)|$ is the volume of the ball (e.g., πR^2 in 2D or $\frac{4}{3}\pi R^3$ in 3D). This provides an unbiased estimate of the source contribution for that step. Combining the boundary jump and the source estimate, we define the overall recursive estimator $\hat{u}(x_k)$ as:

$$\hat{u}(x_k) = \begin{cases} g(\bar{x}_k) & \text{if } x_k \in \partial\Omega_\epsilon \\ \hat{u}(x_{k+1}) + |B(x_k)| f(y_k) G(x_k, y_k) & \text{otherwise} \end{cases} \tag{8}$$

where x_{k+1} is sampled from the sphere boundary, y_k is sampled from the ball volume and $\partial\Omega_\epsilon$ represents the ϵ-shell stopping condition. You can find in our Appendix the udapte of the algorithm.

This framework can be adapted to solve higher-order equations, such as the **Biharmonic equation** ($\Delta^2 u = f$), which appears frequently in elasticity theory and geometry processing. To do so, we decompose the problem into a system of second-order equations:

$$\Delta^2 u = f \iff \begin{cases} \Delta u = v \\ \Delta v = f \end{cases} \tag{9}$$

Here, the auxiliary variable v acts as the source term for u, while v itself is the solution to a Poisson equation driven by f. Algorithmically, this requires a **Nested Walk**: while simulating the walk for u, at each step x_k, we sample a point y_k uniformly within the volume of $B(x_k)$ and launch a new, independent walk to estimate $v(y_k)$. This naive approach results in a quadratic complexity of $O(n^2)$, where n is the average walk length.

4.3 Absorption Term

We can further generalize the solver to include an absorption (or screening) term $\sigma > 0$. This yields the **Screened Poisson equation**, often used in implicit modeling or diffusion-reaction systems:

$$\begin{cases} \Delta u(x) - \sigma u(x) = f(x) & \text{for } x \in \Omega \\ u(x) = g(x) & \text{for } x \in \partial\Omega \end{cases} \tag{10}$$

Physically, the term σ represents a rate of decay or absorption. In the stochastic view, the random walkers are no longer immortal; they have a probability of being "killed" or absorbed as they travel. The Feynman-Kac formula expresses:

$$u(x) = \mathbb{E}_x \left[\int_0^\tau e^{-\sigma t} f(W_t) dt + e^{-\sigma\tau} g(W_\tau) \right] \tag{11}$$

To adapt the Walk on Spheres algorithm, we utilize the recursive integral relation on a ball $B(x)$. Unlike the standard Poisson equation, the Green's function is replaced by the **Yukawa potential** G^σ, and the boundary averaging kernel is replaced by the **Poisson kernel** P^σ for the screened operator:

$$u(x) = \int_{B(x)} G^\sigma(x, y) f(y) dy + \int_{\partial B(x)} P^\sigma(x, y) u(y) d\sigma(y) \tag{12}$$

In this context:

- $G^\sigma(x, y)$ is the Yukawa potential, which decays faster than the harmonic Green's function.

- $P^\sigma(x, y)$ accounts for the fact that the average value on the sphere is no longer equal to the real (due to absorption).

The recursive estimator is modified to include this weighting. If x_{k+1} is sampled from the sphere and y_k from the ball volume:

$$\hat{u}(x_k) = \underbrace{w(R, \sigma)}_{\text{Weight}} \cdot \hat{u}(x_{k+1}) + \underbrace{|B(x_k)| \cdot f(y_k) \cdot G^\sigma(x_k, y_k)}_{\text{Weighted Source}} \quad (13)$$

where the weight $w(R, \sigma)$ (derived from P^σ) depends on the radius and the absorption coefficient. For example, in 3D, this weight is $\frac{R\sqrt{\sigma}}{\sinh(R\sqrt{\sigma})}$. This effectively penalizes longer walks, reducing the influence of distant boundary conditions. A further extension of this framework involves the inclusion of a drift term, corresponding to the **Convection-Diffusion equation**. We provide the mathematical derivation for this case in the Appendix

5 Variable Coefficient PDEs

Real-world physical systems rarely exhibit constant material properties. Phenomena such as heat conduction in composite materials (varying thermal diffusivity), electrical conduction in biological tissues or flow through porous media require solving PDEs with spatially varying coefficients.

5.1 General problem formulation

We consider the general second-order elliptic PDE where the diffusion tensor $\alpha(x)$, drift field $\vec{\omega}(x)$ and absorption coefficient $\sigma(x)$ vary continuously over space:

$$\nabla \cdot (\alpha(x)\nabla u) + \vec{\omega}(x) \cdot \nabla u - \sigma(x)u = f \quad \text{on } \Omega \quad (14)$$

Standard WoS cannot be applied directly here because the harmonic Green's functions $G(x, y)$ and Poisson kernels are only known analytically for operators with constant coefficients. Discretizing these coefficients via FEM would lead to the meshing issues discussed in Section 2.

5.2 The "Fictitious Medium" Strategy

To solve Equation 14 without meshing, Sawhney et al. [5] propose a strategy inspired by *delta tracking* (or null-collision algorithms) used in neutron transport and volume rendering.

The core idea is to transform the variable-coefficient PDE into a **Constant-Coefficient Screened Poisson Equation** by moving all spatial variability into a recursive source term.

(1) **Transformation:** Through a series of variable substitutions (including the Girsanov theorem to handle drift), the original PDE is rewritten as:

$$\Delta U(x) - \bar{\sigma} U(x) = F(x, U) \quad (15)$$

where $\bar{\sigma}$ is a constant bounding parameter (e.g., $\bar{\sigma} = \max(\sigma(x))$).

(2) **Recursive Dependency:** The new source term $F(x, U)$ depends on the unknown solution U itself. While this seems circular, it allows for a recursive Monte Carlo formulation.

5.3 Connection to Volume Rendering

This recursive formulation reveals a deep mathematical isomorphism between solving variable-coefficient PDEs and the **Volume Rendering Equation (VRE)** used in computer graphics.

As shown in the Feynman-Kac formula compared to the VRE:

- **Absorption:** The variable decay $\sigma(x)$ acts like optical density (extinction) in a participating medium.
- **Scattering:** The recursive evaluation of u inside the domain is equivalent to computing in-scattering light.

5.4 Delta Tracking Estimator

The resulting algorithm does not simply jump from sphere boundary to sphere boundary. Instead, it behaves like a photon traveling through a heterogeneous medium. Inside each ball $B(x)$:

- We sample a "free flight" distance t based on the majorant coefficient $\bar{\sigma}$.
- If the flight ends inside the ball, we treat it as a "null collision". We evaluate the local material properties and probabilistically decide whether to absorb the walker or continue the trajectory.

This "virtual" homogenization allows the solver to adapt to continuously varying coefficients $\alpha(x), \vec{\omega}(x), \sigma(x)$ using only standard ray-tracing operations, without ever creating a volumetric mesh.

6 Acceleration and Variance Reduction

6.1 Importance Sampling

In the standard WoS estimator for the Poisson equation, the volume integral $\int_{B(x)} f(y)G(x, y)dy$ is estimated by sampling y uniformly. However, if the source term $f(y)$ or the Green's function $G(x, y)$ varies significantly, uniform sampling leads to high variance (noise).

Sampling the Green's Function. The harmonic Green's function $G(x, y)$ has a singularity at x. Sampling points y near x more frequently reduces variance. In 2D, instead of uniform sampling, we sample the radius r from a distribution proportional to $rG(r)$, and the angle θ uniformly. This neutralizes the singularity in the estimator variance.

Sampling the Source. If the source $f(x)$ is sparse (e.g., point sources or localized heat), uniform sampling will miss it most of the time. We can construct a probability density function (PDF) $p(y) \propto |f(y)|$ to guide samples toward high-energy regions.

$$F_{IS} = \frac{f(y)G(x, y)}{p(y)} \quad (16)$$

For complex scenarios, we employ Multiple Importance Sampling (MIS), a technique from rendering. MIS computes a weighted combination of samples drawn from both f (source) and G (geometry), ensuring robustness regardless of which term dominates.

6.2 Control Variates for WoS

To estimate the integral of a function $\phi(x)$, we use an approximation function $\tilde{\phi}(x)$ with a known integral c. The estimator becomes:

$$F_{CV} = c + \frac{1}{N} \sum_{i=1}^{N} \left(\phi(X_i) - \tilde{\phi}(X_i) \right) \quad (17)$$

6.2.1 Estimator for the Solution $u(x_0)$. By using a first-order Taylor approximation around x_0, where the linear term integrates to zero

over a sphere ($c = 0$), the improved estimator for the solution is:

$$\hat{u}_{CV}(x_0) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{u}_i(x_0) - \overline{\nabla u}^{i-1}(x_0)\cdot(x_1^i - x_0)\right) \tag{18}$$

where:

- $\hat{u}_i(x_0)$ is the standard WoS estimate from the i-th walk.
- $\overline{\nabla u}^{i-1}(x_0)$ is the gradient's estimate from previous walks.
- x_1^i is the first sample on the sphere $B(x_0)$ for walk i.

6.2.2 Estimator for the Gradient. Similarly, to reduce variance for the gradient, we use the running estimate of the solution $\overline{u}_0^{i-1}(x_0)$:

$$\widehat{\nabla u}_{CV}(x_0) = \frac{n}{R}\frac{1}{N}\sum_{i=1}^{N}\left(\hat{u}_0(x_1^i) - \overline{u}_0^{i-1}(x_0)\right)\nu(x_1^i) \tag{19}$$

where $\nu(x_1^i)$ is the outward unit normal at x_1^i and n is the dimension.

6.3 Denoising

Since Monte Carlo error manifests as high-frequency noise rather than low-frequency bias, it is an ideal candidate for image-space denoising. We apply off-the-shelf deep learning denoisers (such as Intel Open Image Denoise) to the raw PDE solution.

Because the underlying geometry and boundary data are known perfectly (via the BVH), we can provide auxiliary feature buffers (normals, albedo/boundary values) to the denoiser, allowing it to reconstruct sharp features even with very low sample counts (e.g., 4 walks per pixel).

7 Experimental Results

7.1 Convergence Analysis

We first evaluate the convergence behavior of stochastic processes approaching the boundary of the domain. specifically comparing the efficiency of standard Brownian Motion against the Walk-on-Spheres (WoS) algorithm.

To assess computational efficiency, we simulated trajectories using both standard Brownian Motion and the WoS algorithm. As illustrated in Figure 1, a significant disparity in the number of iterations is observed.

The standard Brownian Motion relies on fixed, infinitesimal steps, resulting in a dense trajectory that requires a high number of iterations to reach the boundary ∂D. In contrast, the WoS algorithm leverages adaptive step sizes, corresponding to the radius of the largest inscribed sphere, allowing the process to converge to the boundary in significantly fewer steps. This demonstrates the superior algorithmic efficiency of WoS for boundary hitting problems.

7.2 Laplace and Poisson Equations

To validate our implementation, we addressed the "Polygon Soup" benchmark problem [4]. The geometry is composed of intersecting segments where Dirichlet boundary conditions are defined by RGB color vectors.

Figure 2 presents both the problem configuration and the qualitative results obtained using the Walk-on-Spheres algorithm. As illustrated in the bottom row of the figure, increasing the number

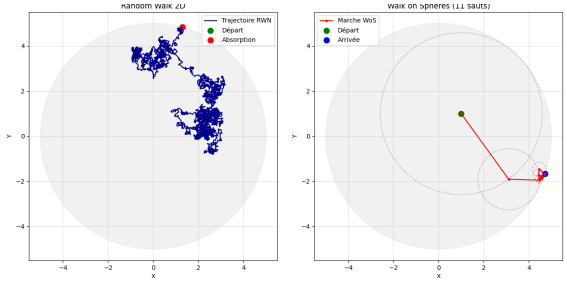
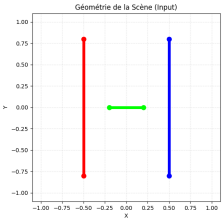


Figure 1: Trajectory comparison between standard Brownian Motion and the Walk-on-Spheres (WoS) algorithm. The WoS method reaches the boundary with substantially fewer intermediate steps compared to the dense path of the Brownian Motion.

of samples (N) progressively reduces the variance (visual noise), leading to a smooth solution.



(a) Problem geometry with RGB boundary conditions.

(b) $N = 20$ **(c) Intermediate** N **(d)** $N = 2048$

Figure 2: Resolution of the Laplace equation on the Polygon Soup geometry. (a) shows the input geometry. (b-d) show the reduction in variance as the number of random walks increases.

Quantitatively, the approximation error diminishes as the sample size grows (Figure 3). This confirms the convergence properties of the Monte Carlo estimator for this boundary value problem.

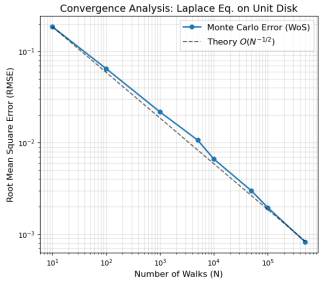


Figure 3: Convergence analysis: Error evolution as a function of the number of samples.

7.2.1 Poisson Equation in 2D. We extended the numerical framework to address the Poisson equation with a non-zero source term. We consider the problem defined on the unit disk D:

$$-\Delta u = f \quad \text{in } D \tag{20}$$

$$u(x, y) = \cos(2\pi x)\sin(2\pi y) \quad \text{on } \partial D \tag{21}$$

To validate the results against an analytical solution, the source term f is chosen such that:

$$f(x, y) = 8\pi^2 \cos(2\pi x)\sin(2\pi y) \tag{22}$$

The solution is estimated by initiating N random walks from each query point. During the walk, the contribution of the source term f is accumulated at each step using the Green's function for the locally largest inscribed sphere of radius R:

$$G(r) = \frac{1}{2\pi} \ln\left(\frac{R}{r}\right) \tag{23}$$

This allows for the stochastic integration of the source term alongside the boundary contribution.

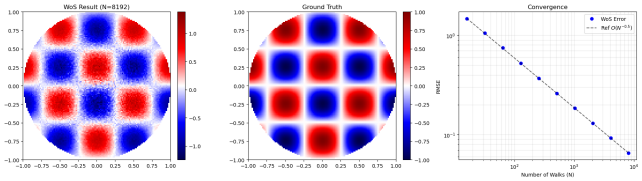


Figure 4: Numerical solution of the Poisson equation on the unit disk using the modified Walk-on-Spheres algorithm.

7.3 Gradient Estimation

Building upon the resolution of the scalar Laplace and Poisson problems, we implemented a numerical estimator for the function's derivatives. This allows for the reconstruction of the gradient field ∇u directly from the boundary data.

Figure 5 illustrates the estimated gradient vectors. The grid-free nature of the method ensures that derivatives can be evaluated at any arbitrary point in the domain without the need for finite difference approximations on a mesh.

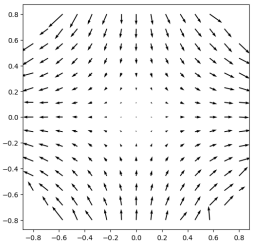


Figure 5: Visualisation of the estimated gradient field.

7.4 Bézier Curves and Shape Deformation

A critical advantage of grid-free geometry processing is its ability to handle domains defined by continuous parametric curves, such as Bézier curves, without rasterization artifacts. We applied the Monte Carlo estimator to problems involving Bézier-defined boundaries, as it offers superior precision for smooth geometries compared to grid-based approximations.

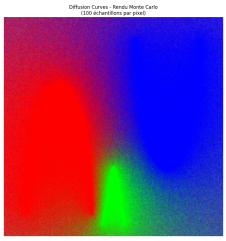


Figure 6: Application of the solver to geometry defined by Bézier curves (Diffusion Curves).

Furthermore, we extended the framework to compute shape deformations. By leveraging the continuous nature of the Monte Carlo integration, we can accurately simulate geometric variations and deformations as shown in Figure 7.

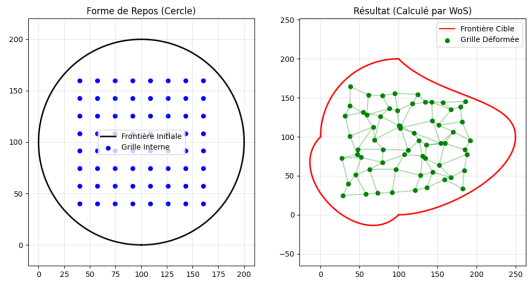


Figure 7: Computation of shape deformation using the Monte Carlo estimator.

7.5 Control Variate and Importance Sampling

A central challenge in Monte Carlo estimation is variance reduction. As previously discussed, several techniques exist to mitigate this issue. In this section, we implement control variates and evaluate their impact. Here you can find the result of our experiment on control variate for the first two experiences :

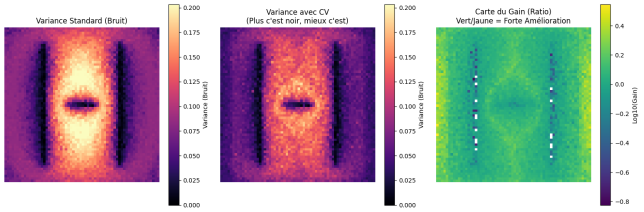
As shown, the variance of the estimator is reduced, which improves the convergence rate. We experimented with a double control variate (targeting both gradient and solution); while this yields superior accuracy, the efficiency gain from a single control variate is already substantial.

7.6 Delta Tracking

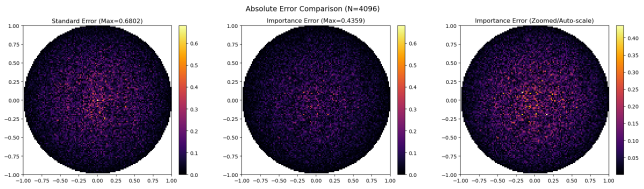
Finally, to evaluate the method on PDEs with spatially varying coefficients, we solved the Screened Poisson equation $\Delta u(x) - \sigma(x)u(x) = 0$ on a square domain with Dirichlet boundary conditions $u|_{\partial\Omega} = 1$.

We defined a procedural absorption coefficient $\sigma(x)$ following a high-frequency checkerboard pattern, alternating between $\sigma = 0$ (free diffusion) and $\sigma = 100$ (strong absorption).

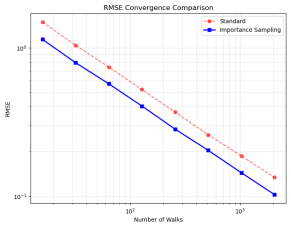
As shown in Figure, the solver accurately reproduces the material structure. In regions of high absorption, the solution decays rapidly



(a) Control Variates (Laplace)



(b) Importance Sampling (Poisson)



(c) Error evolution with Importance Sampling

Figure 8: Analysis of variance reduction techniques. Top row: Visual comparison of solutions. Bottom row: Quantitative reduction in error using Importance Sampling.

to zero, while in free regions, the boundary value diffuses inward. This result confirms that the grid-free approach can effectively simulate heterogeneous materials by relying solely on point-wise evaluation of coefficients.

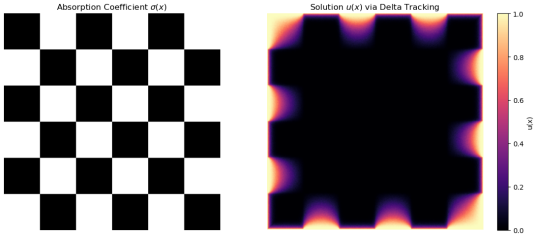
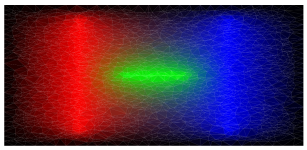


Figure 9: Error linked with Importance Sampling

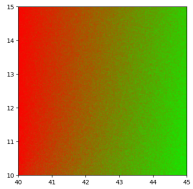
8 Limitations and Advantages

8.1 Advantages

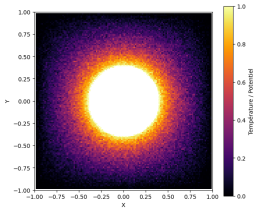
8.1.1 Comparison with a FEM. The main objective of the paper is to highlight the limitations of the FEM when compared to grid-free Monte Carlo approaches. For simple geometries, FEM is often faster and provides a level of accuracy comparable to Monte Carlo estimators. However, the situation changes significantly when the geometry becomes more complex or when high-resolution local information is required.



(a) Laplace with FEM



(b) Zoom on WoS estimations



(c) 3D with WoS

Figure 10: Comparison between FEM and Monte Carlo methods (WoS). (a) FEM solution of the Laplace equation; (b) Illustration of unlimited zooming capabilities with WoS; (c) 3D computation using WoS at slice $z = 0$.

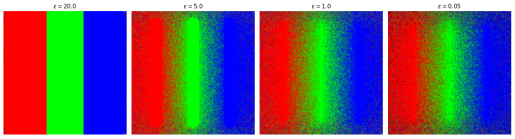
A key advantage of the WoS and Monte Carlo estimators is their ability to refine local regions without any global mesh refinement. By simply zooming on a specific area, the algorithm recomputes local estimates independently of the global domain. This property is crucial for handling highly detailed or irregular geometries, where FEM struggles to generate a sufficiently fine mesh or becomes prohibitively expensive.

Another important limitation of FEM arises in three-dimensional problems. The computational time and memory required to solve a 3D PDE using FEM grow rapidly with the mesh size, often exceeding the capabilities of a standard computer. In our experiments, a moderately refined 3D FEM mesh already required more than fifteen minutes of computation, while the WoS method provided a result almost instantly for a slice at $z = 0$.

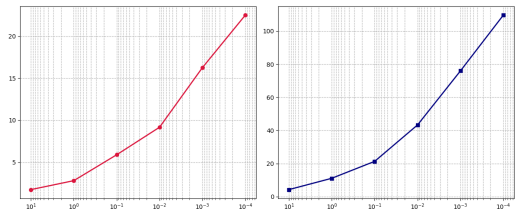
Finally, for complex "polygon soups" containing many intersecting segments or thin structures, FEM may fail to correctly detect or resolve geometric features, leading to inaccurate solutions. In contrast, Monte Carlo methods do not rely on mesh connectivity and naturally handle such configurations. Moreover, while both methods scale with the number of evaluation points N, the computational cost grows as $O(N)$ for FEM whereas it remains essentially independent of the global resolution for WoS.

8.1.2 Parameter ϵ. As discussed earlier, the computational cost associated with the tolerance parameter ϵ scales as $O(\log(1/\epsilon))$. This logarithmic dependence implies that choosing a small value of ϵ does not drastically increase the computation time. On the other hand, selecting a larger ϵ significantly accelerates the algorithm but may introduce a loss of accuracy in the final solution.

A large value of ϵ is particularly useful in other types of experiments. For instance, in the construction of Voronoï diagrams, where



(a) Geometry for different ϵ



(b) Time (right) and number of steps (left)

Figure 11: Influence of the parameter ϵ on geometry, computational effort, and applications such as Voronoï diagram generation.

the objective is to obtain fast approximate partitions rather than exact solutions, a larger tolerance provides a substantial computational advantage while remaining visually acceptable. (see Appendix)

8.1.3 Parameter N. As shown in the previous section, the statistical error decreases as $O(1/\sqrt{N})$, where N is the number of Monte Carlo samples. This guarantees a globally fast and stable convergence: increasing N improves accuracy predictably, although with diminishing returns due to the square-root rate.

8.2 Limitations

8.2.1 Blindness. A significant limitation, discussed by Sawhney et al. [2022], arises from reflecting boundaries. Consider a dumbbell-shaped domain (two chambers connected by a narrow tunnel). The computation time becomes prohibitive due to the stochastic nature of the process. We initially set the tunnel potential to V=0, with the balls acting as heat sources. Contrary to FEM, we observe a loss of accuracy.

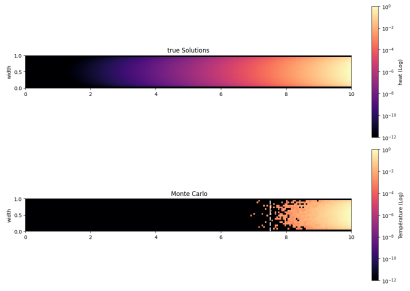


Figure 12: Limit of WoS algorithm

Sawhney et al. [2022] consider reflecting walls, which improves precision but drastically increases computation time as walkers struggle to exit the tunnel. Our implementation required over 5 minutes to converge, whereas FEM required only a few seconds. A potential solution is the "Walk on Stars" method proposed by the same authors in 2023, which uses star-shaped domains to allow for larger steps. We did not explore this extension in this work.

8.2.2 Gradient explosion. Another problem can come with the estimation of the gradient. When a point is close to the boundary, our gradient can explode. It will have some extremes values and it makes it not precise. The algorithm performs well in general but has its own limitations. This instability arises from the gradient estimator, which contains a term proportional to 1/R. In our experiments, we calculated the gradient values to demonstrate the instability near the boundary.

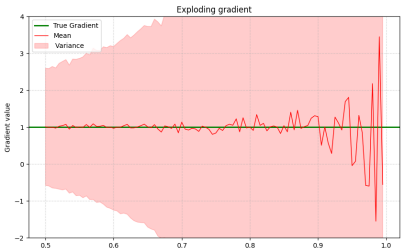


Figure 13: Gradient explosion near of the wall

8.2.3 Perspectives. On our work, we have focused on implementing the different estimators for fixed coefficients. However, a major contribution we can produce is to extend this to varying coefficient. We have explained the theoritical aspect but we have just implemented the delta-tracking and create a drift term.

Moreover, we are limited by the computer's performances because we can't explore more complex geometrical aspect without taking a excessive ressources.

9 Conclusion

This report investigated the Walk on Spheres algorithm as a robust, mesh-free alternative to FEM for solving elliptic PDEs. Our experiments confirm that WoS excels in handling "dirty" geometry (e.g., polygon soups, Bézier curves) and enables infinite local zooming without the memory overhead of global discretization. Nevertheless, limitations remain regarding its universality. We demonstrated significant performance degradation in narrow "tunnel" geometries where random walkers get trapped, as well as variance explosion when estimating gradients near boundaries. Future work should focus on geometric accelerations, such as the *Walk on Stars* method, to overcome these topological constraints while preserving the grid-free advantages.

References

[1] Shizuo Kakutani. 1944. Two-dimensional Brownian motion and harmonic functions. *Proc. Imp. Acad.* 20, 10 (1944), 706–714.
[2] Mervin E Muller. 1956. Some continuous Monte Carlo methods for the Dirichlet problem. *Ann. Math. Statist.* 27, 3 (1956), 569–589.
[3] Karl K Sabelfeld and Nikolai A Simonov. 2016. *Random walks on boundary for solving PDEs.* Walter de Gruyter GmbH & Co KG.
[4] Rohan Sawhney and Keenan Crane. 2020. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Trans. Graph.* 39, 4 (2020).
[5] Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. 2022. Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients. *ACM Transactions on Graphics (TOG)* 41, 4 (2022).

10 Appendix

10.1 Proof of Convergence WoS

Since u is harmonic ($\Delta u = 0$) inside D, it satisfies the **Mean Value Property**. By construction of the algorithm, X_{k+1} is a uniform random variable on the new sphere of radius d_k centered at X_k.

$$u(X_k) = \mathbb{E}[u(X_{k+1}) \mid X_k] \tag{24}$$

This implies that the sequence of random variables $(M_k)_{k\geq 0}$ defined by $M_k = u(X_k)$ is a **discrete martingale**. By the martingale property:

$$u(x_0) = \mathbb{E}[M_0] = \mathbb{E}[M_1] = \cdots = \mathbb{E}[u(X_k)]$$

Let us prove the convergence of the algorithm. Let $d_k = \text{dist}(X_k, \partial D)$. By the definition of the algorithm, the next point is given by:

$$X_{k+1} = X_k + d_k \cdot U_k$$

where U_k is a random vector uniformly distributed on the unit sphere $\mathbb{S}^{d-1}$. Therefore:

$$\mathbb{E}[U_k \mid X_k] = 0 \quad \text{and} \quad \|U_k\| = 1$$

We calculate the conditional expectation of the squared norm:

$$\begin{aligned}\mathbb{E}[\|X_{k+1}\|^2 \mid X_k] &= \mathbb{E}[\|X_k + d_k U_k\|^2 \mid X_k] \\ &= \mathbb{E}[\|X_k\|^2 + 2d_k\langle X_k, U_k\rangle + d_k^2\|U_k\|^2 \mid X_k]\end{aligned}$$

The cross term vanishes because $\mathbb{E}[U_k] = 0$. Thus:

$$\mathbb{E}[\|X_{k+1}\|^2 \mid X_k] = \|X_k\|^2 + d_k^2 \tag{25}$$

Taking the total expectation and iterating the previous relation from $k = 0$ up to N:

$$\mathbb{E}[\|X_N\|^2] = \|X_0\|^2 + \sum_{k=0}^{N-1} \mathbb{E}[d_k^2]$$

Since the domain D is bounded, there exists a radius R such that for every $x \in D$, $\|x\| \leq R$. Thus, the left-hand side is bounded:

$$\|X_0\|^2 + \sum_{k=0}^{N-1} \mathbb{E}[d_k^2] \leq R^2$$

This inequality holds for all N. This implies that the series of expectations converges:

$$\sum_{k=0}^{\infty} \mathbb{E}[d_k^2] < \infty$$

If the series $\sum \mathbb{E}[d_k^2]$ converges, then the general term $\mathbb{E}[d_k^2]$ must tend to 0. Furthermore, by the Monotone Convergence Theorem (implicitly applied here, or more formally, the convergence of bounded martingales applied to $\|X_k\|^2$), we have:

$$\sum_{k=0}^{\infty} d_k^2 < \infty \quad \text{(almost surely)}$$

Therefore:

$$\lim_{k\to\infty} d_k = 0 \quad \text{a.s.}$$

10.2 WoS for Poisson

Algorithm 2 Monte Carlo WoS Solver (Poisson)

Require: Query point x_0, Boundary $\partial\Omega$, Tolerance ϵ, Walks N

$u_{total} \leftarrow 0$ ▷ Accumulator for all walks
for $i \leftarrow 1$ **to** N **do**
  $x \leftarrow x_0$
  $u_{walk} \leftarrow 0$ ▷ Accumulator for current walk
  while $d(x, \partial\Omega) > \epsilon$ **do**
    $R \leftarrow d(x, \partial\Omega)$
    $y \leftarrow \text{RandomPointInBall}(x, R)$
    $vol \leftarrow \pi \cdot R \cdot R$ ▷ Volume term (2D example)
    $u_{walk} \leftarrow u_{walk} + vol \cdot G(x, y, R) \cdot f(y)$
    $x \leftarrow x + R \cdot \text{RandomUnitVector}()$
  end while
  $x_b \leftarrow \text{ClosestPoint}(x, \partial\Omega)$
  $u_{walk} \leftarrow u_{walk} + g(x_b)$ ▷ Add boundary value
  $u_{total} \leftarrow u_{total} + u_{walk}$
end for
return u_{total}/N

10.3 Drift Term (Convection-Diffusion)

Finally, we consider the **Convection-Diffusion equation**, which adds a first-order derivative term representing a drift (or advection) field $\vec{\omega}$:

$$\begin{cases}\Delta u(x) + \vec{\omega} \cdot \nabla u(x) = f(x) & \text{for } x \in \Omega \\ u(x) = g(x) & \text{for } x \in \partial\Omega\end{cases} \tag{26}$$

In the stochastic view, the underlying process is a diffusion with drift. The position of a particle X_t evolves according to the Stochastic Differential Equation (SDE):

$$dX_t = dW_t + \vec{\omega}(X_t)dt \tag{27}$$

where dW_t is standard Brownian motion and $\vec{\omega}$ acts as a velocity vector biasing the walker's trajectory.

To apply the WoS algorithm, we must again adapt the recursive integral formula on the ball $B(x)$. As shown in the literature (e.g., Sabelfeld [3]), the presence of drift changes both the Green's function inside the ball and the probability distribution of the exit point on the sphere.

The recursive formula becomes:

$$u(x) = \int_{B(x)} G^{\vec{\omega}}(x, y) f(y) dy + \int_{\partial B(x)} P^{\vec{\omega}}(x, y) u(y) d\sigma(y) \tag{28}$$

Crucially, the Poisson kernel $P^{\vec{\omega}}(x, y)$ is no longer constant. In the standard Laplace case, the walker exits the sphere uniformly. With drift, the exit probability is concentrated in the direction of $\vec{\omega}$.

- **Biased Sampling:** The distribution of exit points on the sphere corresponds to the **Von-Mises Fisher distribution**. When implementing the walk, x_{k+1} must be sampled from this distribution rather than uniformly.
- **Modified Green's Function:** The internal Green's function $G^{\vec{\omega}}$ is also skewed, meaning the source term $f(y)$ must be weighted differently depending on its position relative to the drift direction.

This modification allows the Monte Carlo framework to simulate fluid flow or transport phenomena without explicit grid-based advection schemes, simply by biasing the random walk steps.

10.4 Voronoï diagram

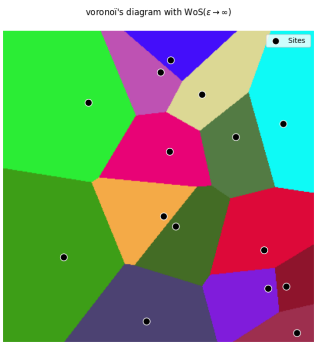


Figure 14: Voronoï diagram obtained with large ϵ