A Monte Carlo Method for Fluid Simulation

DAMIEN RIOUX-LAVOIE*, McGill University and Ubisoft Montreal, Canada
RYUSUKE SUGIMOTO*, University of Waterloo, Canada
TÜMAY ÖZDEMIR, University of Waterloo, Canada
NAOHARU H. SHIMADA, Osaka University, Japan
CHRISTOPHER BATTY, University of Waterloo, Canada
DEREK NOWROUZEZAHRAI, McGill University, Canada
TOSHIYA HACHISUKA, University of Waterloo, Canada

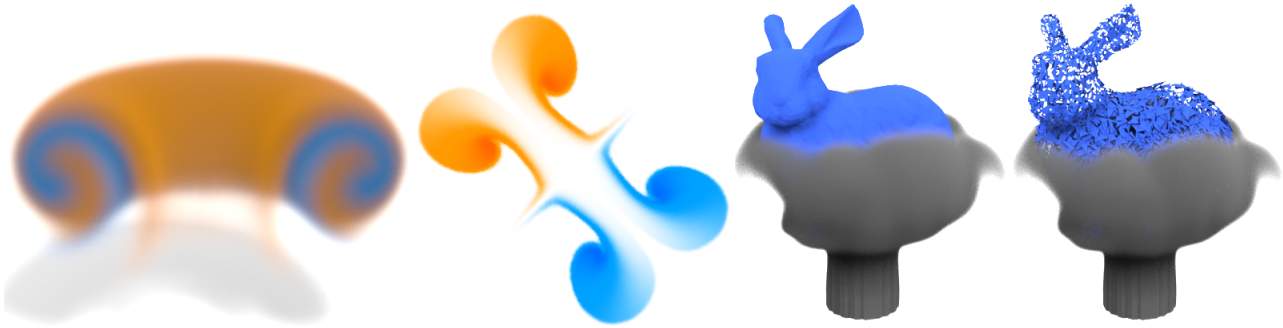


Fig. 1. Our novel Monte Carlo fluid simulator supports 2D and full 3D simulations of viscous incompressible flows by computing pointwise stochastic solutions of the vorticity transport equation. It is easy to implement and treats diverse fluid effects, such as leapfrogging vortex rings (left) and colliding jets (middle). The adoption of a Monte Carlo method to handle boundaries is well-suited to treating nontrivial boundary geometry (right).

We present a novel Monte Carlo-based fluid simulation approach capable of pointwise and stochastic estimation of fluid motion. Drawing on the Feynman-Kac representation of the vorticity transport equation, we propose a recursive Monte Carlo estimator of the Biot-Savart law and extend it with a stream function formulation that allows us to treat free-slip boundary conditions using a Walk-on-Spheres algorithm. Inspired by the Monte Carlo literature in rendering, we design and compare variance reduction schemes suited to a fluid simulation context for the first time, show its applicability to complex boundary settings, and detail a simple and practical implementation with temporal grid caching. We validate the correctness of our approach via quantitative and qualitative evaluations – across a range of settings and domain geometries – and thoroughly explore its parameters' design space. Finally, we provide an in-depth discussion of several axes of future work building on this new numerical simulation modality.

*Authors with equal contribution.

Authors' addresses: Damien Rioux-Lavoie, McGill University and Ubisoft Montreal, Canada, riouxld21@gmail.com; Ryusuke Sugimoto, University of Waterloo, Canada, rsugimot@uwaterloo.ca; Tümay Özdemir, University of Waterloo, Canada, tozdemir@uwaterloo.ca; Naoharu H. Shimada, Osaka University, Japan, NHShimada93@gmail.com; Christopher Batty, University of Waterloo, Canada, c2batty@uwaterloo.ca; Derek Nowrouzezahrai, McGill University, Canada, derek@cim.mcgill.ca; Toshiya Hachisuka, University of Waterloo, Canada, thachisu@uwaterloo.ca.

0730-0301/2022/12-ART240 $15.00
https://doi.org/10.1145/3550454.3555450

CCS Concepts: • **Mathematics of computing** → **Probabilistic algorithms**; **Partial differential equations**; • **Computing methodologies** → **Modeling and simulation**.

Additional Key Words and Phrases: Computational Fluid Dynamics, Monte Carlo Integration, Stochastic Processes, Walk-on-Spheres Algorithm

ACM Reference Format:
Damien Rioux-Lavoie, Ryusuke Sugimoto, Tümay Özdemir, Naoharu H. Shimada, Christopher Batty, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2022. A Monte Carlo Method for Fluid Simulation. *ACM Trans. Graph.* 41, 6, Article 240 (December 2022), 16 pages. https://doi.org/10.1145/3550454.3555450

1 INTRODUCTION

Monte Carlo (MC) methods have been successfully applied to a diversity of problems in, e.g., statistical inference, simulation and integration [Metropolis and Ulam 1949]. In graphics, MC has been used most extensively in physically-based light transport where it outpaced traditional finite element methods to treat challenging radiometric effects with increasingly complex geometric and reflectance models [Kajiya 1986; Pharr et al. 2018]. Recent applications of MC to problems in geometry processing [Sawhney and Crane 2020] further evidence its broader applicability in graphics.

Motivated by this growing adoption, we adapt and demonstrate the utility of MC methods in computational fluid dynamics. We present a new MC approach to generate fluid motions with a pointwise stochastic formulation of solutions to the incompressible Navier-Stokes equations. We leverage a reinterpretation of the deterministic fluid equations as a stochastic process, introducing Monte Carlo integral estimators of the Biot-Savart law that relate fluid vorticity

and velocity fields. This reformulation allows us to define a recursive fluid flow model with striking similarities to simulation and sampling strategies employed in MC rendering.

Concretely, we first devise a recursive numerical integration scheme to solve 2D incompressible Euler equations on open and periodic domains. Next, we develop a stream function-based strategy to enforce free-slip boundary conditions for static or moving obstacles, leading to a Poisson equation that we solve stochastically via Walk-on-Spheres [Muller 1956]. Our solver does not perform any global solve or boundary discretization, enabling a point-wise estimation and treatment of complex boundaries. Our stream function-based approach also allows for the treatment of inflow and outflow conditions on the domain boundary. Lastly, we generalize the method to 3D Navier-Stokes equations (i.e., accounting for viscosity and stretching) using the Feynman-Kac formula that expresses partial differential equation (PDE) solutions as an expectation of a stochastic process, for which we compute with MC.

In its simplest form, our recursive formulation (Fig. 2) exhibits exponential computation complexity (Fig. 3) – akin to how distribution ray tracing scales for the number of light bounces. We overcome this problem with a practical uniform grid-based cache. Our caching scheme further allows us to develop and apply MC variance reduction techniques, including importance sampling of the vorticity field, and a control variate approach that utilizes the vorticity and velocity fields from the previous time step, to increase sample efficiency and accelerate stochastic estimates.

We validate our method against standard grid- and particle-based solvers, exploring and summarizing its behavior under different boundary conditions and parameter settings. The pointwise nature of our method allows an easy parallelization of computation. Our work is the first foray in the space of MC methods applied to fluid simulation in graphics and, in addition to highlighting the current limitations of our proof-of-concept simulators, we outline and discuss a series of open problems associated to scaling MC-based fluid simulators to larger and more challenging flows.

Concisely, our contributions are:

- a novel MC fluid solver using recursive, pointwise probabilistic solutions to the 2D Euler equations based on the Biot-Savart law,
- a walk-on-spheres treatment of inflow, outflow, and free slip solid boundary conditions using stream functions,
- a generalized MC solver for the full 3D incompressible Navier-Stokes equations based on the Feynman-Kac PDE formulation,
- a practical, non-recursive cache-based solver,
- applications of MC variance reduction to fluid simulation, and
- a roadmap of open challenges for scalable MC fluid simulation.

2 BACKGROUND

We review the most relevant work in fluid simulation and MC rendering in computer graphics. We refer readers to comprehensive textbook sources by Bridson [2015] and Pharr et al. [2018] for more details.

Monte Carlo rendering. A pioneering application of MC integration to light transport followed the formalization of the now foundational *rendering equation* [Cook 1986; Kajiya 1986], leading to the basic path tracing algorithm. At the time, finite element radiosity approaches [Goral et al. 1984] were the de facto scheme for solving radiative transport problems in rendering. Since then, MC-based path tracing and its variants have evolved to treat more complex radiometric effects, all while scaling more gracefully with the growing complexity of virtual environments [Pharr et al. 2018]. Today, modern industrial-calibre renderers use path tracing [Pharr 2018] and specialized variance reduction strategies to improve sample the efficiency.

Outside of the vast MC rendering literature, Bowers et al. [2011] proposed to use MC ray tracing to approximate solutions to the Poisson problem for diffusion curves. Their approximation is visually consistent with the solutions to the original Poisson problem in the examples presented. Sawhney and Crane [2020] recently introduced a Monte Carlo framework for solving boundary value PDE problems in computational geometry processing. The *Walk-on-Spheres* (WoS) approach [Muller 1956] they introduced to the graphics community scales well to complex geometries than finite element methods – analogously to the aforementioned evolution of physically-based rendering methods. In a concurrent work, Sawhney et al. [2022] further extended the approach for elliptic PDEs with spatially varying coefficients.

We similarly depart from deterministic solvers and treat fluid simulation with MC, deriving a stream function-based free-slip boundary condition that solves a Poisson equation using a custom antithetic WoS sampler. Our work builds upon the original WoS method by Muller [1956], rather than Sawhney and Crane [2020], since WoS cannot be applied straightforwardly to the Navier-Stokes equations. Furthermore, we propose importance sampling and control variate variance reduction schemes to improve sample efficiency, coupled with a practical caching mechanism to overcome naïve recursive MC estimator complexity. Much like in light transport, allowing the treatment of general scenes without discretization, our MC fluid simulation does not fundamentally require explicit spatial discretization commonly employed in fluid simulation.

Fluid simulation. The numerical prediction of fluid motions plays an important role in a variety of fields, from weather forecasting to the aerospace industry. In computer graphics, there is significant demand to animate flowing water, swirling smoke, flickering flame, and so on. Most work on this topic falls into a few broad families of methods based on discretizing the classical incompressible Euler or Navier-Stokes equations. Eulerian methods assume a static mesh or grid through which the fluid flows and the relevant fluid equations are discretized with finite difference/volume/element ideas [Fedkiw et al. 2001; Foster and Fedkiw 2001; Stam 1999]. Lagrangian methods instead use degrees of freedom that move along with the fluid, often using either meshfree particles such as in smoothed particle hydrodynamics [Desbrun and Gascuel 1996; Müller et al. 2003] or time-evolving meshes that must be adapted when they become too deformed [Clausen et al. 2013; Misztal et al. 2013]. Hybrid methods, which use both Lagrangian particles and Eulerian grids, have become popular because they can offer both the convenience and efficiency of Eulerian grids and the accurate advection of Lagrangian particles [Jiang et al. 2015; Zhu and Bridson 2005].

We rely, in part, on tracing flow characteristics backwards in time for potentially long periods, reminiscent of a family of recent

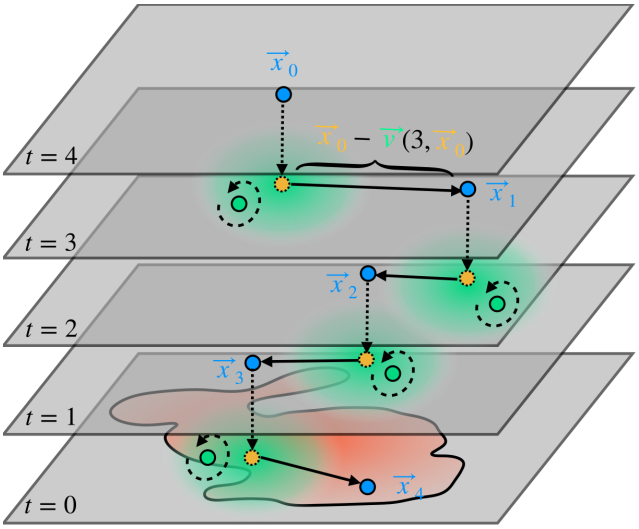


Fig. 2. **MC backtracing:** To compute vorticity at a point and time of interest (in blue, top $t = 4$ layer) we project the point back to the previous time step (yellow) and compute its velocity with our Biot-Savart estimator. To do so, we generate random samples around the projected points (green), recursively computing their vorticity. Once computed, we advect along the velocity to the next position and repeat until we reach the initial condition.

Eulerian methods [Hachisuka 2005; Qu et al. 2019; Sato et al. 2018] that use characteristic mapping [Mercier et al. 2013; Tessendorf and Pelfrey 2011]. In contrast to ours, these methods use an Eulerian grid and advect velocity rather than vorticity.

Another important family of methods replaces the primitive variables (pressure, velocity) with other variables with useful mathematical properties. Vorticity-based methods are a common example and can likewise be treated in either an Eulerian or Lagrangian fashion. In the Eulerian setting, Mullen et al. [2009] presented a circulation-preserving schemes viewed through the lens of discrete differential geometry Stream functions (2D) or vector potentials (3D) are closely related and defined such that velocity is the curl of a potential, and hence inherently incompressible. Bridson et al. [2007] used stream functions to design procedural flows; Ando et al. [2015] used vector potentials to animate liquids with incompressible bubbles. More broadly, stream function-vorticity methods have a long history in computational fluid dynamics [Campion-Renson and Crochet 1978; Peeters et al. 1987] and recent developments seek Eulerian, non-primitive approaches using Clebsch variables [Chern et al. 2016; Yang et al. 2021]. Our method tracks vorticities as the main primitive variables.

Vortex particle methods. A Lagrangian treatment in vorticity variables leads to the vortex particle method, first introduced by Chorin [1973]. The particles represent moving point sources of vorticity, and the velocities needed to move the particles are reconstructed via the Biot-Savart law (Eq. 4). The textbook by Cottet and Koumoutsakos [2000] provides a thorough introduction. Early adaptations of such ideas to computer graphics used both vortex particles [Park and Kim 2005] and vortex filaments [Angelidis and Neyret 2005]. Subsequent developments have considered vortex sheets [Brochu et al. 2012; Pfaff et al. 2012] and vortex segment clouds [Xiong et al. 2021]. The computational complexity of naive vortex methods is $O(N^2)$, where

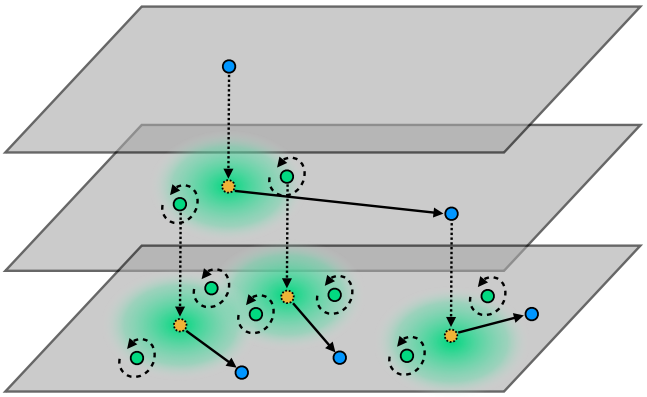


Fig. 3. **Exponential complexity:** Applying our basic recursive formulation leads to exponential cost as, starting from a point of interest (blue, top), we must trace back for every random sample generated during the Biot-Savart approximation (green), leading to an exponential branching factor.

N is the number of vortex elements. Several attempts have been made to improve their computational efficiency, such as the vortex-in-cell method [Couët et al. 1981], fast multipole method [Greengard and Rokhlin 1987], and PPPM [Zhang and Bridson 2014].

Our work also exploits the vorticity form of the fluid equations and the Biot-Savart law to derive a new recursive integral formulation that is amenable stochastic MC estimation. Doing so enables a wide variety of MC acceleration and/or variance reduction methods, among which we present importance sampling and control variates.

Probabilistic fluid models. There has been some research into probabilistic treatments of the Navier-Stokes equations: the *Fourier transformation method* [Jan and Sznitman 1997], the *Lagrangian flow method* [Constantin and Iyer 2007] and *FBSDS method* [Busnello et al. 2005; Delbaen et al. 2015] to name a few. For a more in-depth review, we refer readers to the work of Cruzeiro [2020]. However, this body of work primarily aims at the mathematical study of the Navier-Stokes equations, including questions of existence and uniqueness; no practical numerical simulation has been conducted to the best of our knowledge. We partially use this body of work – most directly that of Busnello et al. [2005] – to construct a stochastic, numerical approach to the Navier-Stokes equations that properly treats viscosity and vortex stretching using the Feynman-Kac formula. Sawhney et al. [2022] used this formula as a basis to transform time-independent varying coefficients PDE to constant coefficients. We instead use it to handle the time-dependent Navier-Stokes equations. In a quite different vein, there has been a recent resurgence of effort in graphics on Lattice Boltzmann methods [Li et al. 2018, 2020], including complex obstacle support [Lyu et al. 2021]. These methods employ probability density functions to model the evolution of fluid particles as a kind of cellular automaton, and are highly parallelizable. While both being stochastic methods, our method is not directly related to Lattice Boltzmann methods.

3 BASIC METHOD

Fluid motion is commonly modeled by the incompressible Euler or Navier-Stokes equations describing the evolution of the velocity field $\vec{v}$ in time. By considering the vorticity field $\vec{\omega} \equiv \nabla \times \vec{v}$, one can instead derive a time-evolution equation for vorticity [Cottet

and Koumoutsakos 2000] that is the basis of various Lagrangian, Eulerian, and hybrid fluid solvers [Elcott et al. 2007; Park and Kim 2005; Selle et al. 2005]. We adopt this vorticity-based approach, albeit departing substantially from the typical formulations.

Vorticity advection is determined uniquely by a flow's velocity and vorticity fields. In contrast to Lagrangian vorticity methods which propagate discrete vorticity elements forward in time, we will use a semi-Lagrangian-like interpretation that conceptually traces "backwards in time" along the flow to determine the vorticity at a query point. To determine the required paths through the flow at any instant in time, one can reconstruct the velocity field from the vorticity field according to the Biot-Savart law. We will apply MC to this problem. The combination of backward tracing and MC yields our base recursive MC estimator of fluid flow (see Figure 2.)

3.1 Vorticity Equation

Given a velocity field $\vec{v}$ and corresponding vorticity field $\vec{\omega}$, we start with the well-established vorticity transport equation [Cottet and Koumoutsakos 2000] for incompressible flow in three dimensions:

$$D\vec{\omega}/Dt = (\vec{\omega}\cdot\nabla)\vec{v} + \nu\nabla\cdot\nabla\vec{\omega}\,, \tag{1}$$

where ν is the kinematic viscosity coefficient and $D/Dt = \partial/\partial t + (\vec{v}\cdot\nabla)$ is the material derivative. We assume constant density and viscosity, no external forces (for now), and we defer discussion of boundary conditions and external forces. Temporarily assuming zero viscosity and considering the case of a 2D fluid, the so-called vortex stretching term $(\vec{\omega}\cdot\nabla)\vec{v}$ is also zero, so it simplifies to

$$D\omega/Dt = 0\,, \tag{2}$$

which corresponds to pure advection of vorticity. Furthermore, the absence of a third dimension reduces the vorticity vector field to a scalar field $\omega = \frac{\partial}{\partial x}v_y - \frac{\partial}{\partial y}v_x$. This advection depends implicitly on the velocity field, which is in turn determined by the vorticity: the apparent simplicity of this evolution equation can thus deceptively misrepresent the fact that it models much more than a simple/passive "one-way" advection process.

In what follows, we denote the dependence of field quantities on time t and position $\vec{x}$ with parentheses, e.g., $\vec{\omega}(t,\vec{x})$. and we use the following notational conventions in 2D: $\nabla\times\Psi = \left[\partial_y\Psi, -\partial_x\Psi\right]$, $\Psi\times\vec{A} = \left[-\Psi A_y, \Psi A_x\right] = -\vec{A}\times\Psi$ and $\nabla\times\vec{A} = \partial_x A_y - \partial_y A_x$, where Ψ is a scalar field and $\vec{A} = \left[A_x, A_y\right]$ is a 2D vector field.

3.2 Semi-Lagrangian Approach

Analytical solutions to (2) are not available in the general case. One popular family of numerical methods to solve such advection problems is *semi-Lagrangian* schemes [Fedkiw et al. 2001; Stam 1999]. After discretizing in time with time step Δt, a basic semi-Lagrangian scheme with Forward Euler time integration of trajectories approximates the updated vorticity $\vec{\omega}(t,\vec{x})$ at position $\vec{x}$ by querying the previous time step vorticity field $\vec{\omega}(t-\Delta t,\vec{x})$ as

$$\vec{\omega}(t,\vec{x}) \approx \vec{\omega}(t-\Delta t, \vec{x} - \vec{v}(t-\Delta t,\vec{x})\Delta t). \tag{3}$$

3.3 Biot-Savart Law

We can advect the vorticity field $\vec{\omega}(t,\vec{x})$ based on (3) given the velocity field $\vec{v}(t,\vec{x})$, which we in turn derive from $\vec{\omega}(t,\vec{x})$ using the *Biot-Savart law* [Cottet and Koumoutsakos 2000],

$$\vec{v}(t,\vec{x}) = \int_{\mathcal{X}} \vec{\omega}(t,\vec{y})\times\vec{G}(\vec{x}-\vec{y})\,\mathrm{d}\vec{y}\,, \tag{4}$$

where $\mathcal{X}$ is a domain on which $\vec{\omega}$ is non-zero and the kernel is $\vec{G}(\vec{x}-\vec{y}) = (\vec{x}-\vec{y})/(2\pi|\vec{x}-\vec{y}|^2)$ in 2D and $\vec{G}(\vec{x}-\vec{y}) = (\vec{x}-\vec{y})/(4\pi|\vec{x}-\vec{y}|^3)$ in 3D. The velocity field obtained from (4) satisfies both the divergence free property ($\nabla\cdot\vec{v} = 0$) and the definition of vorticity ($\vec{\omega} = \nabla\times\vec{v}$). Closed form solutions to (4) are usually unavailable and we choose to estimate it with MC.

3.4 Monte Carlo Integration

MC integration is a stochastic numerical method to approximate integrals as a carefully weighted average of independent random samples. We make use of this method to approximate the Biot-Savart integral (4) to compute the velocity $\vec{v}(t,\vec{x})$.

Biot-Savart MC estimator

Let $\{\vec{y}_i\}_{i=1}^{n_{\mathrm{mc}}}$ be n_{mc} independent samples drawn from probability distribution $p(\vec{y}\,|\,t,\vec{x})$. The MC estimator of (4) is

$$\langle\vec{v}(t,\vec{x})\rangle = \frac{1}{n_{\mathrm{mc}}}\sum_{i=1}^{n_{\mathrm{mc}}}\frac{\vec{\omega}(t,\vec{y}_i)\times\vec{G}(\vec{x}-\vec{y}_i)}{p(\vec{y}_i\,|\,t,\vec{x})}\,. \tag{5}$$

3.5 Monte Carlo Fluids

Building atop these aforementioned ideas, our central goal is to calculate the vorticity $\omega(\bar{t},\bar{x})$ of a dynamic fluid at simulation locations $\bar{x}$ and times $\bar{t}$. The fluid has initial conditions $\omega(0,\vec{x})$, $\vec{x}\in\mathbb{R}^2$. Any subsequent state $\omega(\bar{t},\bar{x})$ is necessarily a function of the initial state $\omega(0,\vec{x})$ since the vorticity equation is deterministic. We begin by expressing this evolution function explicitly.

Recursive integral formulation. Combining the semi-Lagrangian scheme (3) for time evolution and the Biot-Savart law (4), we represent $\omega(\bar{t},\bar{x})$ as an integral formulation of $\omega(\bar{t}-\Delta t,\bar{x}')$ at a previous time step:

$$\omega(\bar{t},\bar{x}) \approx \omega(\bar{t}-\Delta t,\ \bar{x} - \vec{v}(\bar{t}-\Delta t,\ \bar{x})\Delta t) \tag{6}$$

$$= \omega\Bigg(\bar{t}-\Delta t,\ \underbrace{\bar{x} - \Delta t\int_{\mathcal{X}}\omega(\bar{t}-\Delta t,\ \vec{y})\times\vec{G}(\bar{x},\ \vec{y})\,\mathrm{d}\vec{y}}_{\bar{x}'}\Bigg). \tag{7}$$

We can represent $\omega(\bar{t}-\Delta t,\ \bar{x}')$ as an integral equation of $\omega(\bar{t}-2\Delta t,\ \bar{x}'')$, backtracing recursively in time until reaching the initial condition $\omega(0,\ \vec{x})$. That is, $\omega(\bar{t},\ \bar{x})$ can be represented as a multiply-nested recursive integral formulation of $\omega(0,\ \vec{x})$. A similar multiply-nested recursive formulation is used in rendering to derive the *path space* integral formulation of light transport [Pharr et al. 2018].

Monte Carlo backtracing. We can now replace the analytical integral in (7) with the MC estimator (5) to solve the recursive integral by tracing backwards in time to the initial conditions. While we discretized the time axis similarly to previous approaches in fluid simulation, this estimator provides a point-wise estimation of velocity without explicit discretization in space.

Consider an illustrative MC backtracing scenario in Figure 2. For simplicity we assume that $\Delta t = 1$. In order to compute the vorticity at

a point of interest $\vec{x}_0$ (blue) and a given time $t = 4$ (top), we compute the velocity at this location in the previous timestep (yellow) with our Biot-Savart Monte Carlo estimator. Doing so requires us to generate random samples around the projected point (green) and compute their vorticities recursively (Fig. 3). Upon completing the recursion, we advect the point of interest to its backward position by the velocity and proceed recursively until we reach the initial condition of the system (at $t = 0$).

Discussion. To our knowledge, MC backtracing is the first numerical method to present a *pointwise* solution of the fluid equations both in space and time. Given the benefits of MC – namely, the ability to trade spatial discretization errors typical of alternative solvers for variance in the MC estimate and the capability to apply various variance reduction techniques - MC backtracing may be an attractive alternative. Moreover, when combined with physically-based Monte Carlo rendering methods (e.g., path tracing), the variance in the fluid solver *and* the renderer may likely lead to more perceptually pleasant errors [Cook 1986]. One important caveat in the formulation described thus far is that the recursive evaluations of the vorticity at each backtraced sample lead to an *exponential* computational complexity (Fig. 3). This major problem is implicitly woven into our formulation but we will devise solutions that alleviate it.

4 ADVANCED METHODS

We present several extensions of our basic method to tackle a broader range of fluid problems and address computational constraints. We first introduce support for free-slip solid boundary conditions through an adaptation of the walk-on-spheres algorithm using a stream function formulation. We then extend our formulation and method from the 2D Euler equations to the fully 3D incompressible Navier-Stokes, using the Feynman-Kac reformulation of the solution as a forward/backward stochastic process. Finally, we devise a more practical caching-based extension of the MC backtracing algorithm that overcomes the exponential cost of the base algorithm.

4.1 Boundary Conditions

When no boundary conditions are mentioned, we assume infinitely far or periodic boundaries and apply the Biot-Savart velocity estimator (5). To handle free-slip (or no through) solid boundary conditions, we employ a Poisson equation (with Dirichlet boundary conditions) to convert vorticity into a *stream function*, the curl of which yields the desired velocity.

Free-slip boundaries. We present our derivations in the 2D setting for clarity, but our ideas generalize fairly naturally to 3D (Section 4.2) using vector potentials in place of stream functions [Bridson et al. 2007]. We initially consider boundaries with zero normal motion, i.e., $\vec{v}_{\text{wall}} \cdot \hat{n} = 0$; under free-slip conditions, solid tangential velocity has no effect on the flow so we need not constrain it. We define the stream function $\Psi(t, \vec{x})$. We flip the usual sign convention for the stream function ($\vec{v} = \nabla \times \Psi$) for convenience. (e.g., [Bridson et al. 2007]) such that

$$\vec{v} = -\nabla \times \Psi. \tag{8}$$

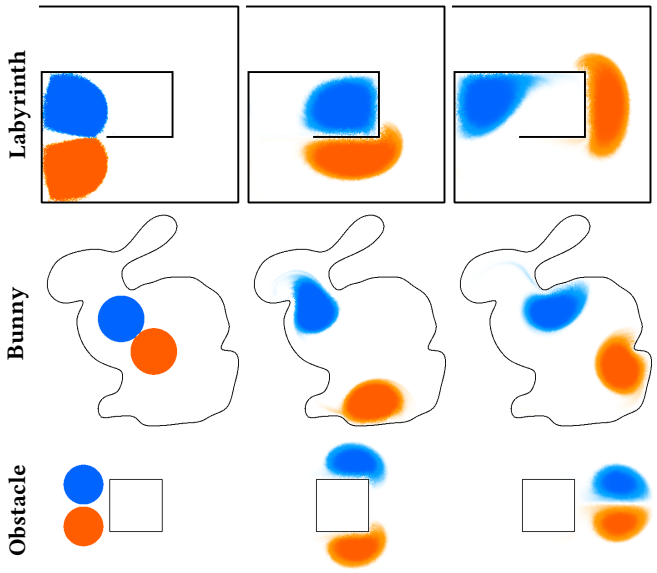


Fig. 4. **Free-slip boundary conditions:** Left to right, $t = 0, 5, 10$ seconds. Two vortices propagating in a maze (top), a flow inside a more complex bunny-shaped boundary defined using a signed distance function (middle), and vortices flowing around a solid obstacle in a periodic domain (bottom).

Combining this expression with the definition of vorticity gives the following relationship (in 2D):

$$\omega = \nabla \times \vec{v} = \nabla \times (-\nabla \times \Psi) = \nabla \cdot \nabla \Psi \,. \tag{9}$$

From (8), if Ψ is constant along the boundary (i.e., an isosurface of Ψ), the normal component of the velocity will be zero (i.e., $\vec{v} \cdot \hat{n} = 0$). Moreover, if the boundary is connected and piecewise smooth, we can arbitrarily set the isovalue to 0 without loss of generality (Fig. 4). With multiple disconnected boundaries, we can still set the same isovalue to each of them if we can expect zero net flow between the boundaries. Thus, after solving the Poisson equation (9) for such a stream function we obtain a velocity field that satisfies our desired boundary condition. In 3D, the true boundary condition $(\nabla \times \Psi) \cdot \hat{n}$ is more involved as it features derivative interactions among multiple components of the vector Ψ [Ando et al. 2015]. Like Bridson et al. [2007], our current 3D results simply use $\Psi = \vec{0}$; further study will be needed to properly resolve scenarios with multiple objects and nontrivial topologies.

Free-Slip Boundary Conditions

Given a domain $D \subset \mathbb{R}^2$, we impose slip boundary conditions using a stream function Ψ of the velocity field $\vec{v} = -\nabla \times \Psi$ by solving the following Poisson equation with Dirichlet boundary conditions:

$$\nabla \cdot \nabla \Psi(\vec{x}) = \omega(\vec{x}) \quad \text{for } \vec{x} \in D \text{ and} \tag{10}$$

$$\Psi(\vec{y}) = g(\vec{y}) \quad \text{for } \vec{y} \in \partial D, \tag{11}$$

where $g(\vec{y}) = 0$ and ∂D is the boundary of D.

Note that we do *not* first compute a current full-domain vorticity field and then solve this Poisson equation as a distinct step; rather, we use WoS to tightly integrate it into our recursive algorithm as a direct replacement for the Biot-Savart MC velocity estimator (5).

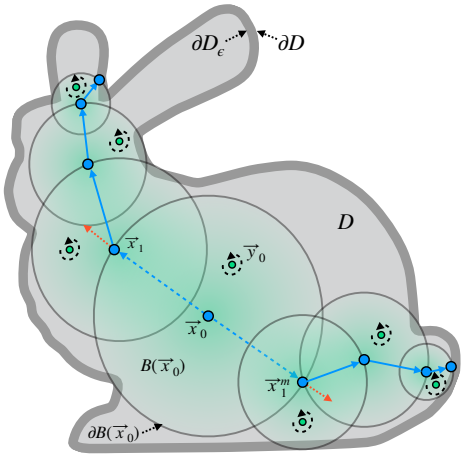


Fig. 5. **Walk-on-Spheres gradient computation visualization:** consider the interior samples $\vec{y}_k \sim \partial B(\vec{x}_k)$ (green) and boundary samples $\vec{x}_{k+1} \sim B(\vec{x}_k)$ (blue). We sample two points $\vec{x}_1$ and $\vec{x}_1^m$ on the largest inscribed sphere centered at $\vec{x}_0$ using antithetic sampling and compute their respective normals (red arrows) at the boundary. From these sampled positions, we proceed with a standard walk-on-spheres – tracing two opposing random paths – and end the recursion once we penetrate a threshold region ∂D_ϵ (dark grey) to compute $\hat{\Psi}(\vec{x}_1)$ and $\hat{\Psi}(\vec{x}_1^m)$.

Walk-on-spheres method. The WoS method [Muller 1956] is an MC pointwise solver for linear elliptic PDEs such as Poisson equations with Dirichlet boundaries.Since WoS computes the pointwise solution via MC sampling, it is well-suited to our method.

Our interest is in determining the curl of the stream function (i.e., velocity), so we apply a gradient WoS estimator [Sawhney and Crane 2020] to the aforementioned Poisson equation with Dirichlet conditions, and use the components of $\nabla_{\vec{x}}\Psi$ to form the curl (e.g., in 2D by 90° rotation). To evaluate $\nabla_{\vec{x}}\Psi(\vec{x}_0)$, the WoS algorithm estimates a recursive integral equation by recursively sampling a point $\vec{y}_k$ inside the largest sphere around the current point $\vec{x}_k$ and sampling another point $\vec{x}_{k+1}$ on the boundary of sphere to continue its recursion. A slight improvement is that we use antithetic sampling to get the first boundary sample, meaning that in addition to $\vec{x}_1$ we always add an extra sample, $\vec{x}_1^m$, on the opposite side of the sphere (also applying WoS to it); see Figure 5 for visual intuition. Omitting antithetic sampling can easily lead to a large variance when trying to compute the gradient of a constant non-zero function.

Other boundary conditions. With a slight modification of the construction above, we can support inflow and outflow boundaries. By dividing the solid boundary into distinct pieces, each with a constant stream function value, and connecting them via linearly interpolated stream function values along the inflow/outflow segments of the boundary, we arrive at a stream function along the inflow/outflow whose gradient is parallel to the boundary (Fig. 6). Taking the (negative) curl recovers the velocity that yields the desired perpendicular (constant) inflow or outflow.

We can further generalize our inflow/outflow treatment to moving solid boundaries with prescribed velocities, again with free slip (Fig. 7). In 2D, the stream function along the boundary must satisfy the constraint

$$\vec{v}_\mathrm{b} \cdot \hat{n} = \partial\Psi \,/\, \partial\hat{t}\,, \tag{12}$$

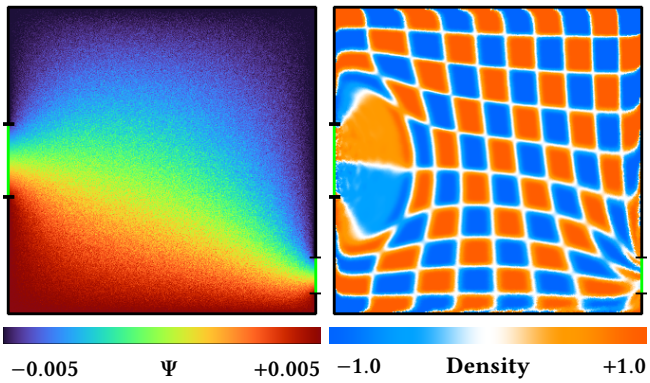


Fig. 6. Inflow and outflow regions highlighted in green on the boundary of the domain. Left: the stream function corresponding to setting the top part of the solid wall to $\Psi = 0.005$ and the bottom to $\Psi = -0.005$. Ψ values at the inflow and outflow vary linearly, yielding constant normal velocity. Right: the motion of an advected checkerboard density field.

where $\vec{v}_\mathrm{b}$ is the velocity of the boundary, and $\hat{n}$ and $\hat{t}$ are the normal and tangent to the boundary.

We can choose any reference point on the solid surface and set it to some constant value Ψ_0, after which determining the stream function along the rest of the boundary (e.g., for a polygon with Ψ_i data at nodes $\vec{s}_i$) from the prescribed velocity field $\vec{v}_\mathrm{b}$ requires integrating the expression above to obtain

$$\psi_{i+1} = \psi_i + \int_{\vec{s}_i}^{\vec{s}_{i+1}} (\vec{v}_\mathrm{b} \cdot \hat{n})\, \mathrm{d}\vec{s}\,. \tag{13}$$

Assuming a divergence-free solid boundary velocity field, the net flux along the surface will be zero; thus, the integration around its boundary loop back to $\vec{x}_0$ will yield the same starting value of Ψ.

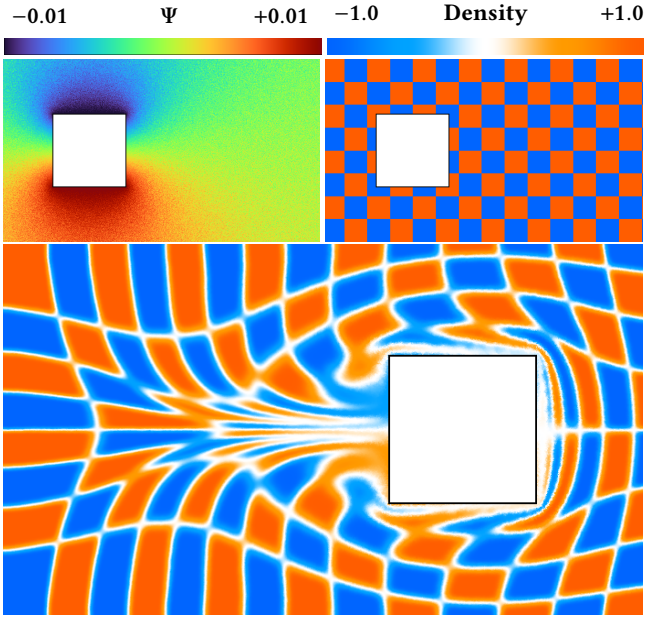


Fig. 7. **Moving obstacles:** A square moves left to right at constant speed through the domain. The stream function (top left) and the initial density field (top right) to be advected, and the result after some time (bottom).

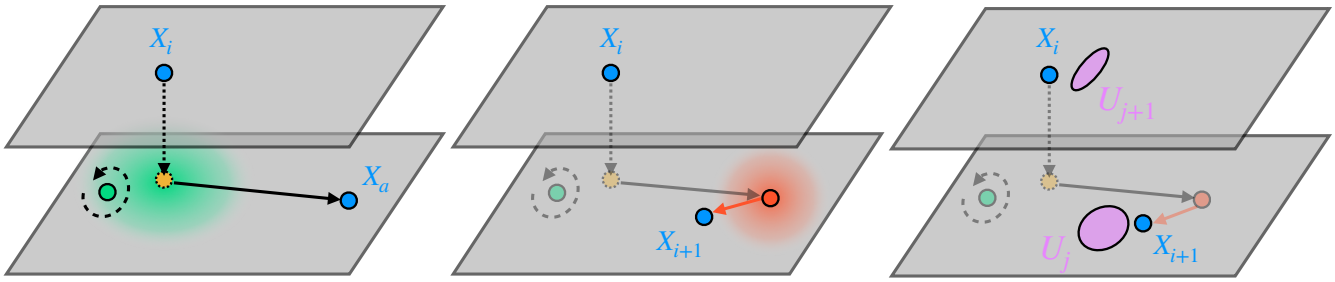


Fig. 8. **Recursive MC Fluid solver for Navier-Stokes equations:** (**Left**) First we advect the position of interest backward in time using a semi-Lagrangian approach and a **MC** velocity estimate. (**Middle**) Then, we simulate viscosity by adding a Gaussian perturbation (with an appropriately-derived variance) to the advected position and compute the vorticity recursively. (**Right**) Finally, we compute and propagate the stretching factor forward in time to the original position by multiplication with the computed vorticity.

For polygons and other simple shapes undergoing simple (e.g., rigid body) motions, we can obtain exact expressions of the stream functions on the obstacle boundaries [Bridson et al. 2007].

4.2 Navier-Stokes Flows

Moving beyond the simple case of flows satisfying the 2D incompressible Euler equation, which we used to provide the intuition behind our backtracing algorithm, we now treat the more challenging case of incompressible Navier-Stokes flows in 3D. That is, we reintroduce the viscous diffusion and vortex stretching terms and rewrite the vorticity transport equation (1) as

$$\frac{D\vec{\omega}}{Dt} = \mathcal{D}_{\vec{v}}\,\vec{\omega} + \nu\nabla\cdot\nabla\vec{\omega}\,, \tag{14}$$

where $\mathcal{D}_{\vec{v}} = (\nabla\vec{v} + \nabla\vec{v}^{\mathsf{T}})/2$ is the strain-rate tensor. The new term $\mathcal{D}_{\vec{v}}\,\vec{\omega}$ is an alternative but equivalent form of the stretching term $(\vec{\omega}\cdot\nabla)\vec{v}$ that will be easier to work with.

Feynman-Kac representation. We wish to compute solutions to (14) at a specific time $\bar{t}$ and position $\bar{x}$, given an initial condition on the vorticity field, analogous to the earlier 2D case. The problem now, however, is more complex: the viscous term effectively adds randomness to the possible trajectories of fluid particles and the stretching term will deform the vorticity according to the flow. To overcome these complexities, we take inspiration from theoretical work on the open problem of the existence and smoothness of solutions to Navier-Stokes [Busnello et al. 2005] and the path integral formulation of quantum mechanics. We apply the Feynman-Kac formula – a mathematical tool that expresses the solution to deterministic PDEs as a stochastic process – to the fluid equations. Doing so will enable us express our numerical solution as a weighted expectation of the initial vorticity, deformed along various trajectories. More formally, the Feynman-Kac formula [Oksendal 2013] states that solutions of the vorticity transport equation satisfy

$$\vec{\omega}(\bar{t},\bar{x}) = \mathbb{E}\left[U_{\bar{t}}\,\vec{\omega}(0,X_{\bar{t}})\,|\,U_0 = I, X_0 = \bar{x}\right], \tag{15}$$

where the expectation is with respect to the Wiener measure, i.e., all possible realizations of the Wiener processes W_s. Here, the *Lagrangian path* X_s is a realization of the time-reversed stochastic process Y_s defined by the stochastic differential equation

$$\mathrm{d}Y_s = -\vec{v}(\bar{t}-s, Y_s)\,\mathrm{d}s + \sqrt{2\nu}\,\mathrm{d}W_s \text{ with } Y_0 = \bar{x}, \tag{16}$$

and the *deformation matrix* along a reversed Lagrangian path X_s is

$$U_t = \exp\left(\int_0^t \mathcal{D}_{\vec{v}}(\tau, X_{t-\tau})\,\mathrm{d}\tau\right), \tag{17}$$

It has been shown that, given a Lagrangian path, this process (17) is a solution to the differential equation

$$\mathrm{d}U_t = \mathcal{D}_{\vec{v}}(t, X_{\bar{t}-t})U_t\,\mathrm{d}t \text{ with } U_0 = I. \tag{18}$$

We refer to (16) as the *backward process*, since it propagates backwards in time (i.e. $s = \bar{t} - t$), and to (18) as the *forward process*. These expressions and their derivations appear in [Busnello et al. 2005] but have not been used to construct a practical numerical method.

While daunting at first glance, the intuition behind these expressions is comparatively straightforward, as illustrated in Figure 8. First, consider the backward process (16) associated to a stochastic differential equation that models the backward trajectory of fictitious particles that cross the position $\bar{x}$ at time $\bar{t}$. The process traces back the trajectories of these particles according to the underlying velocity field (left term, RHS) as we did before, but now additionally accounts for the diffusive process (rightmost term) through the randomness of the Wiener process W_s. All these trajectories carry some vorticity according to their initial state, however these vorticities need also be warped by the local strain-rate tensor along their trajectory to account for vortex stretching. This is precisely what the forward process (18) models: it stretches the initial particle vorticities according to the strain-rate along their trajectory. Finally, the Feynman-Kac formula (15) states that the vorticity at location $\bar{x}$ and time $\bar{t}$ *is exactly* the average over the stretched vorticity carried by all trajectories that crossed the position at the given time[‡].

Monte Carlo estimation. We construct a three-step algorithm to evaluate the expectation (15): an advection step, a diffusion step, and a stretching step. We adopt the notation $X_i = X_{s_i}$ where $s_i = i\Delta s$, $\Delta s = \bar{t}/n$ and $i \in \{0,\cdots,n\}$ for the discretized quantities. We discretize (16) with a semi-implicit variant of the Euler-Maruyama [Maruyama 1955] method, yielding the Lagrangian path

$$X_{i+1} = X_i - \overbrace{\Delta s\,\vec{v}(\bar{t}-s_{i+1}, X_i)}^{\textbf{Advection}} + \overbrace{\sqrt{2\nu\Delta s}\,\vec{\xi}_i}^{\textbf{Diffusion}} \text{ with } X_0 = \bar{x}\,, \tag{19}$$

[‡]An alternative interpretation draws on the Feynman path integral in quantum mechanics, with probability weights assigned to paths and where the quantity of interest at a location is determined by the average over all possible probability-weighted paths.

where $\vec{\xi}_i \sim \mathcal{N}(\vec{0}, I)$ is a normally distributed random sample. Without viscosity this discretization simplifies to that of Section 3. If one uses just a single MC sample during the diffusion step, our treatment of diffusion becomes essentially consistent with that of Chorin [1973], where the position is perturbed using appropriately-scaled Gaussian noise.

Our derivation is actually agnostic to the choice of temporal discretization. For instance, the deterministic term of (16) could instead be discretized using other standard advection approaches, such as Runge-Kutta and MacCormack methods. However, for simplicity of presentation we will use the simple semi-implicit form given in the equations above.

We similarly discretize the deterministic equation (18) with a forward Euler scheme, yielding

$$U_{j+1} = [I + \overbrace{\Delta t \mathcal{D}_{\vec{v}}(t_j, X_{i-t_j})}^{\text{Stretching}}]U_j \text{ with } U_0 = I\,. \tag{20}$$

One can approximate the strain-rate tensor $\mathcal{D}_{\vec{v}}$ in various ways: by finite differences, by applying Monte Carlo integration by passing the gradient of the velocity field through the Biot-Savart integral (4) or through a Hessian WoS estimator, or by converting the vorticity field to a vortex segment representation and advecting it according to the velocity field, as in the work of Zhang and Bridson [2014]. We adopt the last approach due to its superior stability in practice; the term $\mathcal{D}_{\vec{v}}\,\vec{\omega}$ can be approximated by

$$\mathcal{D}_{\vec{v}}\,\vec{\omega}(t,\vec{x}) \approx \frac{|\vec{\omega}(t,\vec{x})|}{h}\left[\vec{v}\left(t, \vec{x} + \overrightarrow{\Delta x}\right) - \vec{v}\left(t, \vec{x} - \overrightarrow{\Delta x}\right)\right], \tag{21}$$

where $\overrightarrow{\Delta x} = (h/2)\vec{\omega}(t,\vec{x})$ and h is the length of the vortex segment. (Note that $\overrightarrow{\Delta x}$ is unrelated to the gradient of $\vec{x}$.) One can estimate the velocities at the ends of a vortex segment using the appropriate velocity estimator to get an estimate of $\mathcal{D}_{\vec{v}}\vec{\omega}$. See Section 4.3 for the implementation details.

Monte Carlo Fluid

When computing the vorticity $\vec{\omega}$ at time $t_i = i\Delta t$ and position $\vec{x}_i$ with initial condition $\vec{\omega}(0,\cdot)$ known, we have

$$\vec{\omega}(t_i,\vec{x}_i) = \frac{1}{n_d}\sum_{j=1}^{n_d}\left[\vec{\omega}\left(t_{i-1}, \vec{x}^{\,j}_{i-1}\right) + \Delta t\left\langle \mathcal{D}_{\vec{v}}\vec{\omega}(t_{i-1}, \vec{x}^{\,j}_{i-1})\right\rangle\right],$$

$$\vec{x}^{\,j}_{i-1} \sim \mathcal{N}(\vec{x}_i - \Delta t\,\langle\vec{v}(t_{i-1},\vec{x}_i)\rangle, \sqrt{2\nu\Delta t}\,I)\,, \tag{22}$$

where n_d is the number of diffusion samples, and $\langle\vec{v}\rangle$ and $\left\langle \mathcal{D}_{\vec{v}}\vec{\omega}\right\rangle$ are Monte Carlo estimates of (5) or (8) and (21).

Implementation. After rewriting the Feynman-Kac expectation (15) as a one-step MC estimate and replacing the velocity and the stretching term with their respective MC estimates, we arrive at our algorithm (see pseudocode in Alg. 1).

4.3 Practical Implementation with Caching

As outlined earlier, a naive implementation of our MC Fluid algorithm (7) suffers from an exponential increase in vorticity samples – all of which must be recursively evaluated – with respect to time (Fig. 3). We offer a strategy to circumvent this problem using a uniform grid cache. Similar to dynamic programming solutions to recursive problems, we store and reuse the vorticity field (and the velocity field in some applications, as we discuss later) as it is computed. We can proceed in two ways.

Algorithm 1: Vor $(t,\vec{x})$: Recursive vorticity computation

Input: Time and position of interest t and $\vec{x}$
Output: Vorticity at $(t,\vec{x})$
Data: I is the Identity matrix
if $t == 0$ **then**
  return InitVor$(\vec{x})$ // Initial condition
$\vec{x}_a \leftarrow \vec{x} - \Delta t$ CompVel$(t-\Delta t, \vec{x})$ // Advection (5) or (8)
$\vec{\omega} \leftarrow \vec{0}$
for $i \leftarrow 1$ ***to*** n_b **do**
  $\vec{x}_i \sim \mathcal{N}(\vec{x}_a, \sqrt{2\nu\Delta t}\,I)$ // Diffusion
  $(\mathcal{D}_{\vec{v}}\vec{\omega})_i \leftarrow$ Stretch$(t-\Delta t, \vec{x}_i)$ // Stretching (21)
  $\vec{\omega} \leftarrow \vec{\omega} + [$Vor $(t-\Delta t, \vec{x}_i) + \Delta t(\mathcal{D}_{\vec{v}}\vec{\omega})_i]$
$\vec{\omega} \leftarrow \vec{\omega}/n_d$
return $\vec{\omega}$

The first is to fill a spatio-temporal grid using the same recursion as before with respect to a position and time of interest. Doing so will gradually fill out a spatio-temporal "cone" of earlier data, rather than evaluating over the whole domain for all time steps. This approach may be ideal when one wishes to evaluate the simulation backwards in time, e.g., in view-dependent simulation.

Alternatively, we can use only a single spatial grid that stores the vorticity computed at the center of each cell at the preceding time step. In essence, at every time step we query this cache to get the previous time step vorticity during Monte Carlo estimation of the Biot-Savart integral for the velocity field. As we will discuss later, this method also conveniently offers a way to generate importance sampled Monte Carlo samples distributed according to the vorticity field, reducing variance and increasing sample efficiency. Unless noted otherwise, all results are generated using this latter approach.

To evaluate the stretching term (21) for 3D simulations, we choose h to be the grid resolution in the absence of boundaries, and in the presence of boundaries we use the minimum of the grid resolution and twice the closest distance to any boundary. For this stretching term computation, we fetch the cached velocities in practice because the bilinear or trilinear interpolation smoothes out the velocity field, which reduces the noise when we compute the stretching term, and because we can save computational time.

4.4 Variance reduction

Our MC approach opens the opportunity to exploit a plethora of *variance reduction methods* within the context of fluid simulation. Indeed, we have already presented the antithetic sampling method for the WoS estimator to reduce the variance. We additionally present a few more basic yet practical variance reduction methods as examples of what would be possible with our MC approach. Section 6 discusses further potential opportunities.

Importance sampling. One of the most popular and basic variance reduction methods is importance sampling. The idea is to generate samples according to a given probability density function (PDF) to

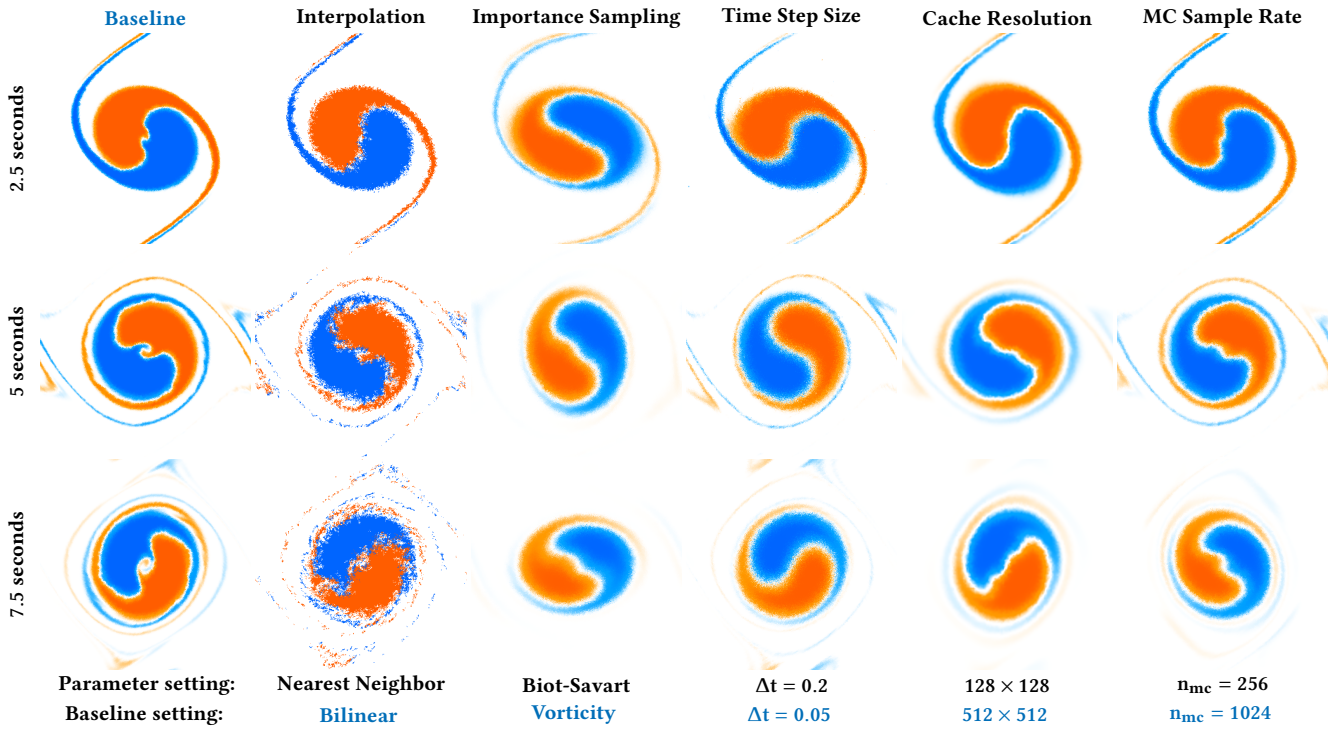


Fig. 9. **Parameter settings:** Top to bottom, frame times are $t = 2.5, 5, 7.5$ seconds. Our baseline simulation (left) with high quality parameter settings (in blue) uses a 512×512 uniform cache resolution, importance sampling of the vorticity field, bilinear interpolation, $\Delta t = 0.05$ seconds, $n_{\text{mc}} = 1024$ samples, and $\nu = 0$. Subsequent columns analyse the impact of adjusting exactly one parameter in favor of a faster but less accurate result. Left to right: baseline/high-qulaity setting, nearest neighbor interpolation, importance sampling the Biot-Savart kernel, larger time step, coarser cache, and lower MC sampling rate.

reduce the variance. A well chosen PDF can drastically reduce the variance, and thus the error of the estimators, and we can come up with a couple of such PDFs that are well-suited to our setting.

The first option is to generate samples proportionally to the integral kernel; the Biot-Savart kernel in the Biot-Savart case (i.e., infinite/periodic boundaries) and the harmonic Green's function in the WoS case (i.e., prescribed boundaries). Sampling according either of those functions is trivial. This importance sampling works especially well for WoS, but not always for the Biot-Savart case.

For the Biot-Savart case, this PDF can be only marginally better than uniform sampling in case non-zero vorticities are concentrated in a small region. The second option in this case is to importance sample directly according to the magnitude of the vorticity field. Since we are already caching vorticities to overcome the exponential cost of recursion, we can utilize this cache to perform importance sampling. Figure 9 illustrates an equal sample comparison between these two options. In this example, the baseline which uses sampling according to vortices produced a more accurate result (e.g., long and thin features of vorticities) than the alternative of sampling according to the Biot-Savart kernel.

Control variates. A sequential nature of fluid simulation fits well to the method of control variates. The method of control variates utilizes another analytically integrable function (a *control variate*) to estimate only the difference between this function and the original integrand via MC integration. This method can reduce variance when the original integrand and the control variate are correlated. In our application, to estimate the velocity using the Biot-Savart law (4), we can use *the vorticity field from the previous time step* as a control variate, as follows:

$$\begin{aligned}\vec{v}(t,\vec{x}) &= \int_{\mathcal{X}} [\vec{\omega}(t,\vec{y}) - \vec{\omega}(t-\Delta t,\vec{y})] \times \vec{G}(\vec{x}-\vec{y})\,\mathrm{d}\vec{y} \\ &\quad + \int_{\mathcal{X}} \vec{\omega}(t-\Delta t,\vec{y}) \times \vec{G}(\vec{x}-\vec{y})\,\mathrm{d}\vec{y} \qquad (23)\\ &= \int_{\mathcal{X}} [\vec{\omega}(t,\vec{y}) - \vec{\omega}(t-\Delta t,\vec{y})] \times \vec{G}(\vec{x}-\vec{y})\,\mathrm{d}\vec{y} + \vec{v}(t-\Delta t,\vec{x}).\end{aligned}$$

Note that having already estimated $\vec{v}(t-\Delta t,\vec{x})$ from $\vec{\omega}(t-\Delta t,\vec{y})$ in the previous time step, this term is available in the current time step. Unlike a typical application of the method of control variates, however, this term $\vec{v}(t-\Delta t,\vec{x})$ is only an estimation with some variance, not an analytical integration of $\vec{\omega}(t-\Delta t,\vec{y})$. When $\vec{v}(t-\Delta t,\vec{x})$ is similarly estimated by using $\vec{\omega}(t-2\Delta t,\vec{x})$ as the control variate, this approach may not reduce variance overall. We circumvent this issue by estimating the initial velocity $\vec{v}(0,\vec{x})$ using a higher sample count than the rest. This approach will propagate variance reduction via this control variate at $t = 0$ all the way to the current time, without increasing the sample count in any other time than $t = 0$.

We generate samples according to the difference of vorticity fields at two consecutive time steps to evaluate the first intergral, and add the cached velocity from the previous time step. Due to the high correlation between the vorticity fields over time, we can expect that the variance of $\vec{\omega}(t,\vec{y}) - \vec{\omega}(t-\Delta t,\vec{y})$ is smaller than $\vec{\omega}(t,\vec{y})$. Thus, this method greatly reduces the variance of our velocity estimate, given that the variance of the initial velocity field is low enough.

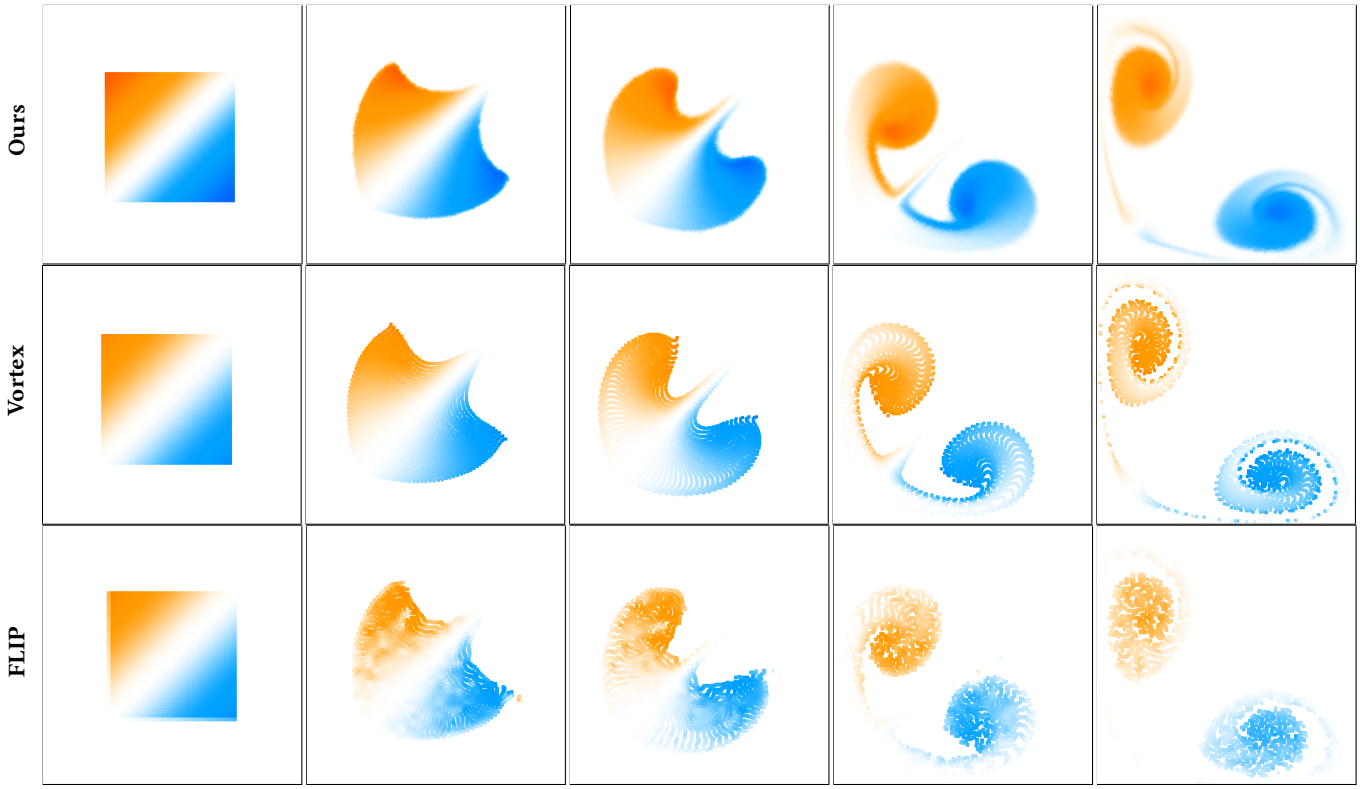
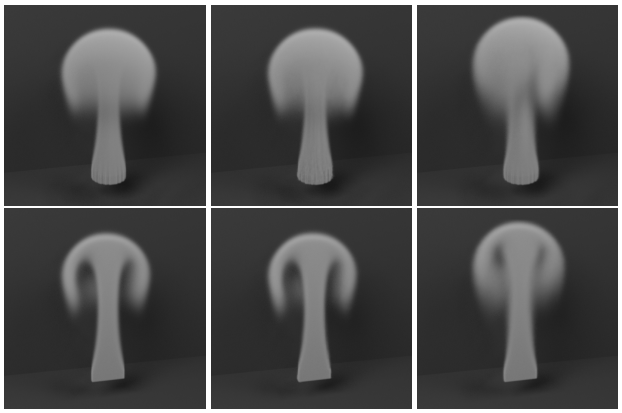


Fig. 10. **Comparison with standard solvers:** Left to right, $t = 0, 1, 2, 5, 10$ seconds. Our method (top) produces results that are consistent with the vortex particle (middle) and hybrid FLIP (bottom) baselines. Colors visualize the signed vorticity at each (particle) position.

Figure 11 illustrates an equal sample comparison when we have the control variate enabled and disabled, and the method of control variates works well as expected. We applied this method only to the 3D scenes (without boundaries), where we expect significant reductions of sampling cost.



(a) reference (b) with control variate (c) w/o control variate

Fig. 11. **Control variate:** When the number of samples used is small, the application of control variate (middle) produces the results closer to reference simulation result (left) compared to the results without control variate (right). We advect a constant density at the bottom according to the velocity field simulated with our method, and render the results fully (top) and with a cross section visualization (bottom).

5 RESULTS

Comparison to existing methods. We compare our method to a pair of representative existing methods (Figure 10): a particle-based solver that implements the vortex particle method of Park and Kim [2005] – using the Biot-Savart law to forward integrate vortex particle trajectories (2500 particles) while treating free slip boundaries with a panel method (80 panels) – and a (hybrid) grid-based FLIP gas solver [Bridson 2015; Zhu and Bridson 2005] with grid resolution $n_x = 50$, 6 FLIP particles per grid cell, and $\Delta t = 0.05$. We visualize the vorticity field, in blue (positive) and orange (negative). We initialize the methods with the same vector field. Our simulation output is in strong agreement with the two baselines, suggesting that it indeed generates a valid solution to the fluid equations.

Our method using WoS runs at an average of 21 seconds per frame (with a multi-threaded CPU implementation). Both the vortex particle method and FLIP run orders of magnitude faster (i.e., roughly 0.005 and 0.25 seconds respectively). Sawhney and Crane [2020] also noted a similar performance gap in their comparisons between WoS and finite-element methods for geometry processing. Despite this significant room of improvement in computation time, our method does have some fundamental advantages. First, the lack of reliance on a linear solve and the point-wise nature of our method make it easily parallelizable and GPU friendly (as evidenced by our ShaderToy and CUDA implementations, discussed below). Second, particle methods rely on nearest neighbor search to interpolate values which can become quite expensive with many particles, which

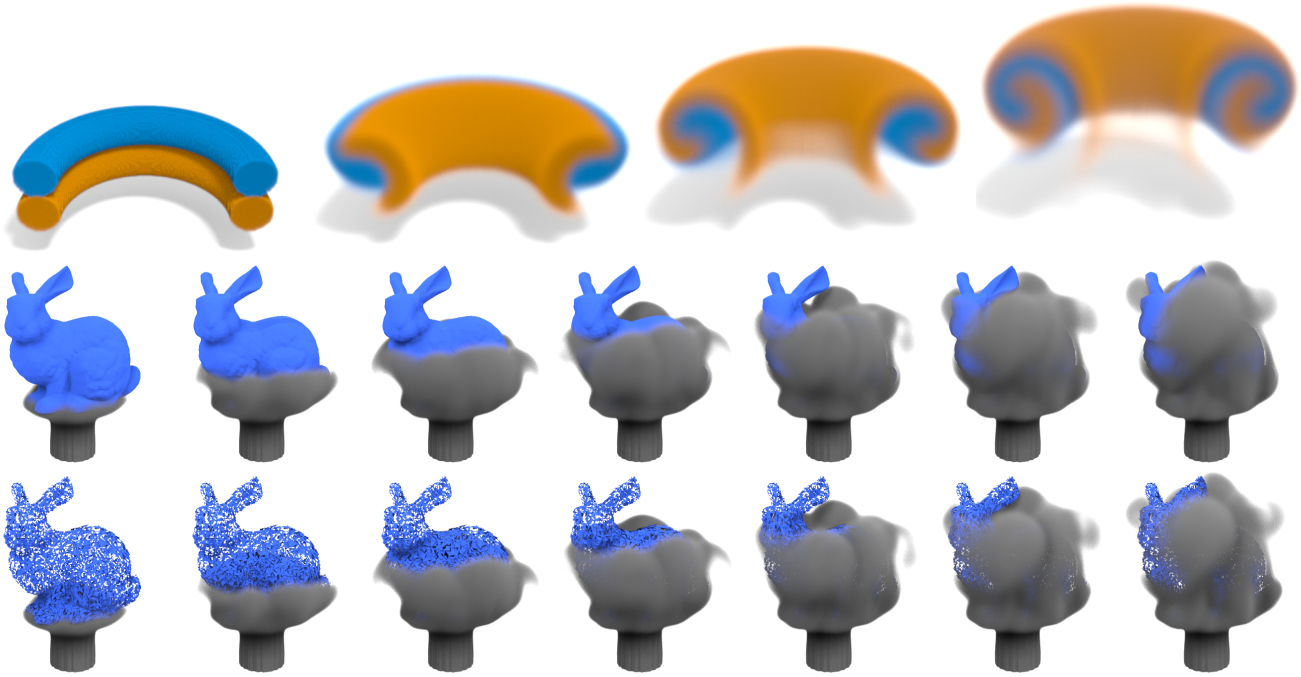


Fig. 12. We simulate two leapfrogging vortex rings (top) with a cross section visualization to illustrate the interior flow. We passively advect constant density fields toward closed mesh (middle) and triangle soup Stanford bunnies (bottom). We render all simulations with Blender's [2022] principled volume shader.

does not exist in our method. Third, grid-based solvers are generally suffering from numerical diffusion, as evidenced in the fluctuations of colors in Figure 10, which is less evident in our method due to its point-wise nature. Last, our formulation based on WoS enables handling of noisy boundaries with no additional effort, as we discuss later. Similar to its application in geometry processing [Sawhney and Crane 2020], a further study and better implementation of our MC approach would reduce the performance gap, leaving the fundamental advantages of MC as noted above.

Viscosity. The viscosity formulation of our 3D flow solver (Section 4.2) applies equally well in 2D. Figure 13 presents a 2D viscous simulation using our approach and a 512×512 uniform cache, nearest neighbor interpolation, importance sampling of the vorticity cache,

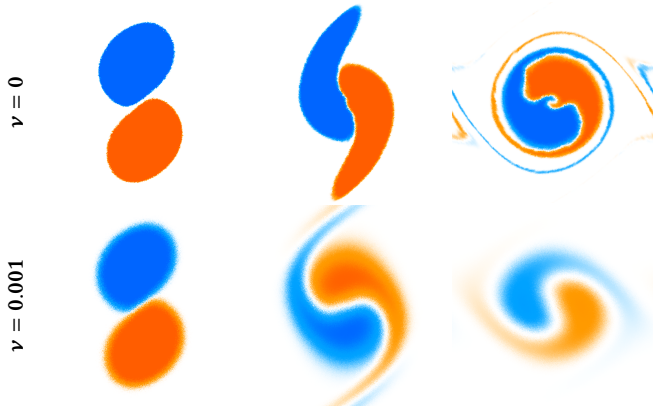


Fig. 13. **Viscous simulation:** Left to right, $t = 0.2, 2, 5$ seconds. Inviscid simulation with $\nu = 0$ (top) and viscous with $\nu = 0.001$; $n_d = 4$. (bottom)

$\Delta t = 0.05$, and $n_{mc} = 1024$. Given that physical diffusion has an effect similar to interpolation, we found nearest neighbor (rather than bilinear) interpolation to be acceptable here.

Boundary conditions. Figure 4 presents various solid boundary configurations. All three scenes use a uniform cache of 512×512, $\Delta t = 0.05$, $n_{mc} = 256$, and $\nu = 0$. As our method for free-slip boundary conditions relies on WoS, it benefits from some of the same attractive properties as the method of Sawhney and Crane [2020], e.g., flexibility in the choice of geometric representations, such as meshes, polygon soup, or even *unsigned* distance functions. In Figure 6, we illustrate our inflow/outflow boundary conditions. Since the integral of the stream function on the boundary is zero, the corresponding velocities yield zero net flux; that is, exactly as much matter enters the domain as exits.

Figure 7 illustrates our moving boundary support. Here, we treat a constant translational velocity for simplicity, since this imposes a constant stream function on the boundary and simplifies computation. Obtaining analytic formula for complex rigid bodies undergoing simple motion is a possibility for more advanced applications, as noted in Sec. 4.1. Both the inflow and moving boundary application were implemented as high-performance real-time demos in the online GPU ShaderToy framework, evidencing the suitability of our approach to parallel computation.

Figures 12 (bottom) and 14 show that our method can accept complex geometry without the added effort of other methods [Azevedo et al. 2016; Hyde and Fedkiw 2019; Lyu et al. 2021], such as challenging tetrahedral mesh or cut-cell generation for velocity-based schemes or panel methods in vorticity schemes. As emphasized by Sawhney and Crane [2020], accurate conforming mesh generation for a complex obstacle can take many minutes or hours; however,

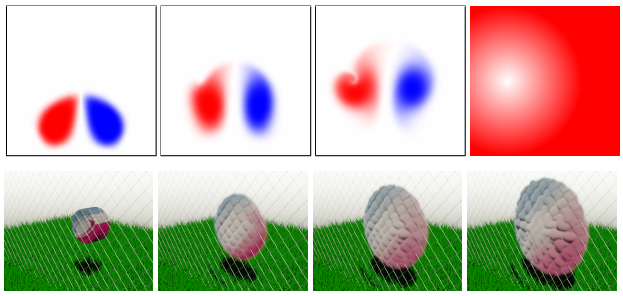


Fig. 14. **Subgrid-size obstacle:** Our approach can take into account a tiny obstacle. A subgrid-sized square (the rightmost image visualizes its unsigned distance function) affects the fluid flow (top). In 3D, subgrid-width chain link fence wires affect the fluid flow (bottom).

as long as we have access to a method for fast evaluation of the distance field, our Monte Carlo-based methods can immediately be applied. In the triangle soup boundary example in Figure 12, we construct a BVH tree of triangles to evaluate distance to the closest boundary. Figure 12 shows a case when the domain contains an obstacle whose size is smaller than the cache grid resolution (about 1/39 of a grid cell width for the 2D scene, and 1/3 of a cell width for the 3D scene), yet is naturally respected by the flow. A typical grid-based solver would either miss this geometry altogether, rely on conservative rasterization to nonphysically inflate the obstacle, or need to introduce an approximate drag model to influence the flow. Notably, the complex 3D fence topology of 14 exceeds the limits of what the simplified $\Psi = \vec{0}$ boundary condition is expected to accurately support, leading to somewhat more damped flow from one side to the other. Extending WoS to support the $\Psi = \nabla\phi$ boundary condition of Ando et al. [2015] may be one avenue to address this shortcoming.

Convergence. In the limit as Δt goes to zero and the number of samples approaches infinity, our method formally should converge to the correct solution (up to the accuracy limited by cache, when applicable). Figure 10 qualitatively demonstrates that our method indeed gives a result that is consistent with existing techniques.

We opted to evaluate the convergence of our method using the steady-state inviscid Taylor-Green vortex flow [Taylor and Green 1937], for which a closed-form solution is known. Figure 15 shows log-scale plots of the root mean square error (RMSE) against the number of Biot-Savart samples n_{mc}, number of cache cells $n_{\text{c}} = n_{\text{x}}^2$ and time step interval Δt. All errors were computed at the physical times $t = 0.25, 0.5, 1$ using importance sampling of the vorticity, bilinear interpolation, RK4 advection, without control variate. We also let $\Delta t = 2^{-6}$, $n_{\text{x}} = 2^{10}$ and $n_{\text{mc}} = 2^{10}$ whenever they are fixed. A well-known fact about MC methods is that their error diminishes in inverse proportion to the square root of the number of samples, i.e. $O(1/\sqrt{n_{\text{mc}}})$ [Robert and Casella 1999]. We confirmed this behavior in our experiment. Similarly, we observed $O(1/\sqrt{n_{\text{c}}})$ which is reasonable given our use of bilinear interpolation. These trends are observed only when the other parameters are good enough to have negligible error contributions. Finally, we observed a convergence order of approximately $O(\Delta t^{0.85})$. However, convergence is lost whenever Δt gets too small. This regime change is in line with the use of a semi-Lagrangian advection scheme [Xiu and Karniadakis 2001]. Intuitively, the error stops decreasing when the interpolation error starts dominating the advection error. Since more steps are needed to reach the same physical time, the error increase as Δt gets smaller.

3D simulations. We demonstrate our method in 3D, with and without obstacles (Figure 12) using CUDA parallel GPU implementations. The leapfrogging simulation uses a uniform cache resolution of 256^3, trilinear interpolation, the control variate method enabled, $\Delta t = 0.1$, $n_{\text{mc}} = 128$, and $\nu = 0$. The initial velocity field is estimated with $n_{\text{mc}} = 16384$, importance sampling the initial vorticity field. The simulation time is roughly 3.5 seconds per step on a machine with two NVIDIA Tesla P100 GPUs at these settings. The Stanford bunny obstacle with slip boundaries uses a uniform cache resolution of 128^3, trilinear interpolation, importance sampling the harmonic Green's function, $\Delta t = 0.1$, 128 WoS paths with maximum 8 steps, 1024 samples for the volume integral evaluation (Eq. 14 in the paper by Sawhney and Crane [2020]), and $\nu = 0$. Simulation time is roughly 70 seconds for the triangle mesh boundary and 75 seconds for the triangle soup boundary per step on a machine with four NVIDIA Tesla V100 GPUs using BVHs for closest distance computation. We believe this small difference of runtimes (additional 5 seconds for triangle soup) is because of the additional cost for the closest distance evaluation.

We perform no low-level simulator optimization, focusing instead on the method's feasibility; GPU memory access optimizations and faster BVH traversal implementations can be explored. We also

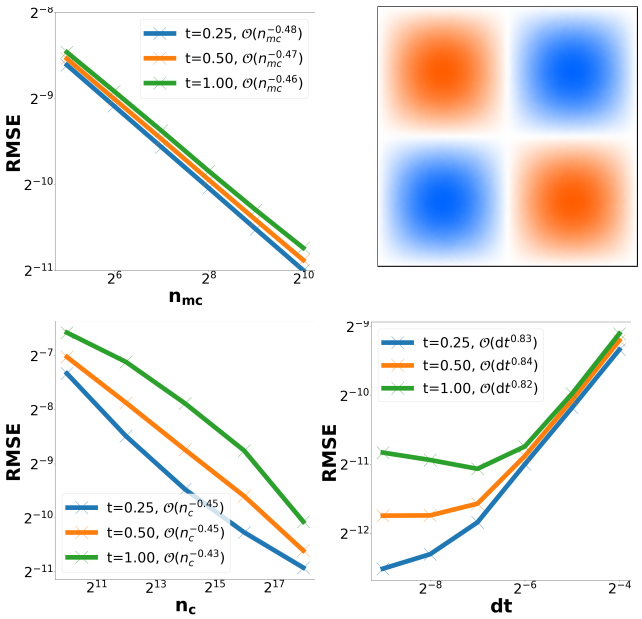


Fig. 15. **Convergence analysis: Root mean squared error computed against the analytical solution $\omega(x, y) = \sin(x)\cos(y)$ for $x, y \in [-\pi, \pi]^2$ (top right). We show the convergence profile of different parameters at different physical times $t = 0.25, 0.5, 1$ seconds. We observe that our method does converge numerically to the true solution as the number of cell $n_{\text{c}} = n_{\text{x}}^2 \to \infty$ (bottom left), when the number of samples $n_{\text{mc}} \to \infty$ (top left) and when Δt is not too large or small (bottom right).**

foresee and briefly discuss many interesting open avenues of work (Section 6) to improve performance, and scale to larger simulations drawing inspirations from the rendering community.

Control variates. The method of control variates as described in Sec. 4.4 can greatly reduce the variance of simulation. Figure 11 shows the effect of our control variate. All three simulations use a cache resolution of 128^3, trilinear interpolation, $\Delta t = 0.1$, and $\nu = 0$. The reference simulation (a) uses importance sampling according to the Biot-Savart kernel with $n_{\text{mc}} = 2048$. The simulation with control variate (b) uses the control variate strategy in Sec. 4.4 with $n_{\text{mc}} = 32$, with the initial velocity field estimated with $n_{\text{mc}} = 16384$ using importance sampling according to the initial magnitude of the vorticity field. The simulation without control variate (c) uses importance sampling according to the magnitude of the vorticity field with the same $n_{\text{mc}} = 32$. We can observe that even with as few as 32 samples per query point, the control variate approach produces a result very close to the reference while simply importance sampling the vorticity field leads to large deviations in the smoke motion due to the extensive noise in the estimated velocity field.

Parameter settings. To better understand the impact of the parameters of our solver on simulation output and performance, we illustrate the effects of the various parameters in Figure 9. The first column is a reference simulation using our method with reasonable high-quality parameter settings: a uniform cache resolution of 512×512, importance sampling based on the vorticity field, bilinear interpolation, $\Delta t = 0.05$ seconds, $n_{\text{mc}} = 1024$ samples, and $\nu = 0$. All other columns modify a single parameter *in a manner that purposefully degrades simulation quality*. The results thus show how changing each parameter can degrade the solution.

Due to MC's stochastic nature, nearest neighbor interpolation produces noisier (i.e., higher variance) results that compound over time. Since bilinear interpolation yields smoother results and is nearly free on, e.g., modern GPU architectures, it is the obvious choice. We also compare two non-trivial MC importance sampling functions to reduce the variance of the estimator and, as expected, importance sampling from the vorticity field yields sharper and more accurate results than sampling according to the Biot-Savart kernel (see Section 4.4) since the latter decreases slowly and omits information of the current simulation state. The discrepancy here is particularly strong when the vorticity is sparsely distributed across the domain. While our advection scheme is unconditionally stable, using time steps that are too large or a cache resolution that is too low results in inevitable loss of detail in high variation regions. Finally, since standard MC error decreases at a rate of $O(1/\sqrt{n_{\text{mc}}})$, the number of samples required for high quality results can be high; when we reduce the sample count by a factor of four, we note a significant loss of details.

6 DISCUSSION AND FUTURE DIRECTIONS

Our work is the first step towards a new family of fluid solvers based on Monte Carlo methods. We demonstrated the potential of our MC-based fluid solvers to generate high quality and accurate results, but there remains much to be explored; a better understanding on how we can best exploit the unique properties, and new trade-offs of noise and computation time within fluid animation to name a few general directions. Echoing Sawhney and Crane [2020], open directions include geometric robustness and flexibility, parallelism, adaptivity and output sensitivity, reduced (or eliminated) dependence on mesh discretization, and further reduction of variance. Below we discuss limitations and propose avenues of future work.

Control variate. The control variate we used in Eq. 23 is not the general formula for control variates. The general form uses a mixture between the estimator and control function with a control parameters. It would be interesting to study how one could introduce an optimal control parameter in our setting to further reduce the estimator variance.

Bias. Currently, our method has two sources of bias – the first one coming from the cache interpolation error and the other from the semi-Lagrangian advection – resulting in energy loss as time advances. The first source, e.g. the cache interpolation bias, is not inherent to our fundamental formulation. One way this could be resolved, without exponential cost, is through the use of a shared particle cache (and sampled locations for MC integration) if one allows recursion to go back to the initial conditions. However, the second source, e.g. the advection bias, is intrinsic to our advection process and solving this issue remains an open question. It comes from the fact that the nonlinearity of our advection scheme (3) does not commute with the MC estimator, much like how naive transmittance estimation through ray marching is bias [Novák et al. 2018]. Further down the road, it would be interesting to explore whether the work of Misso et al. [2022] can be used in our context to generate an unbiased estimator.

Semi-Lagrangian dissipation. For tracing semi-Lagrangian trajectories, we used either forward Euler or RK4, and as expected observed somewhat improved results from the latter option. The dissipation that remains comes from interpolating from the cache, similar to grid-based methods; using higher order interpolation therefore ought to result in less dissipation. It would be interesting to look at incorporating more advanced advection or time-splitting schemes, such as MacCormack, BDF2, advection-reflection , IVOCK, etc., to further lower the dissipation.

Boundary conditions. Investigating a broader set of boundary conditions, such as no-slip – the main source of vorticity in viscous flows – and mixed boundaries is another interesting direction. While some approaches exist [Maire and Tanre 2013; Simonov 2017], effectively handling Neumann and mixed boundaries in WoS is still an open problem, as discussed, for example, by Sawhney et al. [2022].

In general, our free-slip boundary condition framework is only effective for a single closed and connected boundary; otherwise, one needs to know the difference in stream function isovalues between the separate boundaries. Similarly, a 3D vector potential is further complicated by its nontrivial null spaces; however, successfully generalizing to multiple disjoint objects could benefit from MC's ability to handle far more complex geometry, as explored by Sawhney and Crane [2020]. Exploring applications of the Kelvin transform [Nabizadeh et al. 2021] in an MC fluid context would enable efficient simulations on infinite domains, with or without obstacles.

Fig. 16. Fluid simulation results using a uniform cache (left) and an adaptive cache (middle) using one-twentieth memory footprint that of the uniform one. Slowly varying regions are sparsely cached while dense sampling occurs in regions exhibiting steep or discontinuous variations.

Besides inflow/outflow boundaries, we assume no external forcing or sourcing, but a generalization of the Feynman-Kac theorem can be used to incorporate these effects [Busnello et al. 2005]. Extensions to liquids are more challenging, as they require new free surface boundary conditions [Lundgren and Koumoutsakos 1999] and a deforming surface representation.

Advanced caching methods. We have presented how adopting a uniform grid cache avoids exponential cost and makes the method feasible in practice. However, our framework is not limited to a uniform grid and we can consider several different alternatives, each with associated tradeoffs. We believe that adaptive caching methods or a data structure similar to particle-based/hybrid methods could all be viable alternatives. A detailed study is beyond the scope of our initial exploration of MC for fluid animation. Nevertheless, we comment on two possible alternative implementations with preliminary examples.

The first alternative is an adaptive grid cache, replacing the uniform grid with an adaptive tree structure to exploit sparsity. Due to our method's pointwise estimate feature, we can reduce the storage cost for caching *without* affecting the MC estimator itself; other methods typically require modifications to their numerical algorithms such as discrete gradient computation, for example. Figure 16 illustrates our adaptive cache prototype result. It exhibits a large reduction in memory, although this prototype does not yet offer a significant speed gain.

Another exciting alternative uses a set of scattered (unstructured) cache points. Such points could be formed from Lagrangian particles, e.g. particles advected along the simulation, Eulerian, e.g. that are not advected, or even both. Lagrangian point caches can be constructed by importance sampling the particles with respect to the initial vorticity strength, ensuring that particles are distributed proportionally to the vorticity field at all time. Using this type of cache makes our MC approach applicable also to a *particle-based solver*, albeit based on an entirely different mathematical framework than the existing particle-based solvers. Figure 17 shows that this method gives a consistent result with our grid cache-based method.

Caches are also used in rendering to accelerate the computation of multiple bounces of light [Jarosz et al. 2008; Krivánek et al. 2005; Ward et al. 1988]. Since our equations have a similar structure as a recursive equation in rendering, more careful allocation of cache points and reconstruction in our problem can also be beneficial. Those prior work in rendering, however, would not be directly applicable to our method since fluid dynamics and light transport have different characteristics in terms of estimation errors (e.g., the split-sphere model in radiance caching [Ward et al. 1988] has no clear role in our framework).

Alternatives to caching. Our cache-based implementation, and any alternative caching methods discussed in the previous subsection, result in additional bias beyond that introduced by time discretization. This issue motivates a desire to explore and devise alternative solutions for resolving the exponential cost of our base recursive formulation. This exponential cost, in its essence, is similar to the problem of having an exponential cost in path tracing [Kajiya 1986] if we were to split a path at each bounce (i.e., exponential to the number of bounces). Path tracing avoids this exponential growth by tracing along only *one* direction per bounce. While we have empirically found that this approach is not applicable in our setting, some further study of MC methods may avoid this exponential cost.

Quantum methods for numerical integration [Shimada and Hachisuka 2020] may be one potential avenue due to the exponential nature of computation via quantum bits. Russian roulette [Arvo and Kirk 1990], an unbiased method to terminate recursive MC processes according to their expected contribution, can be applied to "early-out" fluid trajectories that contribute negligibly to the final dynamics. Similarly, path (resp. trajectory) splitting [Vorba and Křivánek 2016] can be used to adaptively refine our sampling.

Variance reduction. While the control variate approach we presented for the Biot-Savart case is quite effective, we have only begun to explore the broader landscape of variance reduction methods in MC. Designing an efficient sampling strategy in WoS would be crucial to make the method more practical in the presence of solid boundaries. Compared to the Biot-Savart case, the design space to explore for effective importance functions for WoS is even more flexible. For the boundary integration in WoS, it would be interesting to investigate how directional importance functions, such as the von-Mises Fisher or circular Cauchy distributions [Fisher 1995], could be leveraged. Drawing inspiration from the field of light transport, we can employ multiple importance sampling to combine *several potentially good* PDFs into more robust and sample efficient strategies. Reusing the sample WoS paths within each iteration or even over consecutive time steps similarly to ReSTIR [Bitterli et al. 2020] could also be an interesting direction. Since our approach is based on MC, it is compatible to Markov chain MC methods [Brooks 1998] and significant variance reduction akin to Metropolis light transport [Veach and Guibas 1997] might be possible in our MC fluid solver. A concurrent work on the bidirectional WoS method [Qi et al. 2022] could be combined with our method.

Tighter coupling with rendering. Perhaps most excitingly, the similarities of our method to MC-based light transport simulations suggest that a tighter integration between fluid and light transport simulations may prove fruitful. For example, since our fluid simulation generates errors that manifest as noise, and since possibly

Fig. 17. An alternative implementation with Lagrangian particle cache (right) yields results consistent with our baseline uniform grid (left).

only a subset of the domain is rendered from any individual camera setting, one may be able to adapt the simulation sampling rate to account for the precision needed by the rendering algorithm. Last-mile denoisers of rendered images [Chaitanya et al. 2017; Kalantari et al. 2015], further diversify this design space and trade-offs therein. For example, one might be able to design a denoiser which removes noise from *both* rendering and simulation (via our method) in a coupled manner.

Relatedly, heterogeneous volumetric rendering algorithms [Novák et al. 2018] build acceleration structures and variance reduction techniques to accelerate their convergence, and these structures can inform (and be informed) by MC estimates of the underlying fluid dynamics. For example, one could imagine a hybrid approach in which a coarse grid-based simulation is used to guide the rendering process and high resolution local resimulations are performed on the fly using our approach when the renderer requires higher-resolution fluid density data.

7 CONCLUSION

Motivated by the success of Monte Carlo methods in light transport and their more recent introduction to geometry processing, we proposed a Monte Carlo framework to solve for fluid motion. Our solver generates stochastic pointwise solutions to the incompressible Navier-Stokes equations and is based on the Feynman-Kac formula applied to the vorticity transport equation of fluid flow. We developed a simple and parallelizable implementation, leveraging a cache to counter the exponential cost of a naive treatment of our recursive estimator formulation. We further extended our method to treat boundary conditions for free-slip moving solids, inflows, and outflows, using a stream function formulation and the Walk-on-Spheres algorithm to treat the associated Poisson problem, which enables simulations with complex boundary shapes with minimal effort. We applied three variance reduction techniques – antithetic sampling, importance sampling, and control variate – to the Monte Carlo estimators to accelerate the computation. We demonstrated the practical feasibility of this new perspective, providing a realization of our framework that is novel, easy to implement, and effective. As the first such numerical solver, and a major departure from standard approaches, our work highlights the unique properties, current limitations, and exciting future directions of MC fluid animation.

ACKNOWLEDGMENTS

This research was partially funded by NSERC Discovery Grants (RGPIN-2018-05669 & RGPIN-2020-03918 & RGPIN-2021-02524) and a grant from Autodesk. The bunny mesh is courtesy of the Stanford Computer Graphics Laboratory.

REFERENCES

Ryoichi Ando, Nils Thuerey, and Chris Wojtan. 2015. A Stream Function Solver for Liquid Simulations. *ACM Transactions on Graphics (TOG)* 34 (2015). https://doi.org/10.1145/2766935

Alexis Angelidis and Fabrice Neyret. 2005. Simulation of Smoke based on Vortex Filament Primitives. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*.

James Arvo and David Kirk. 1990. Particle transport and image synthesis. *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*.

Vinicius C Azevedo, Christopher Batty, and Manuel M Oliveira. 2016. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

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 Trans. Graph.* 39, 4, Article 148 (jul 2020), 17 pages. https://doi.org/10.1145/3386569.3392481

John C. Bowers, Jonathan Leahey, and Rui Wang. 2011. A Ray Tracing Approach to Diffusion Curves. In *Proceedings of the Twenty-Second Eurographics Conference on Rendering* (Prague, Czech Republic) *(EGSR '11)*. Eurographics Association, Goslar, DEU, 1345–1352. https://doi.org/10.1111/j.1467-8659.2011.01994.x

Robert Bridson. 2015. *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press.

Robert Bridson, Jim Houriham, and Marcus Nordenstam. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics (TOG)* 26, 3 (2007).

Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Citeseer.

Stephen Brooks. 1998. Markov chain Monte Carlo method and its application. *Journal of the royal statistical society: series D (the Statistician)* 47, 1 (1998), 69–100.

Barbara Busnello, Franco Flandoli, and Marco Romito. 2005. A probabilistic representation for the vorticity of a three-dimensional viscous fluid and for general systems of parabolic equations. *Proceedings of The Edinburgh Mathematical Society* 48 (2005). https://doi.org/10.1017/S0013091503000506

Anne Campion-Renson and Marcel J Crochet. 1978. On the stream function-vorticity finite element solutions of Navier-Stokes equations. *Internat. J. Numer. Methods Engrg.* 12, 12 (1978).

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (jul 2017), 12 pages. https://doi.org/10.1145/3072959.3073601

Albert Chern, Felix Knöppel, Ulrich Pinkall, Peter Schröder, and Steffen Weißmann. 2016. Schrödinger's smoke. *ACM Transactions on Graphics (TOG)* 35, 4 (2016).

Alexandre Joel Chorin. 1973. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics* 57 (1973).

Pascal Clausen, Martin Wicke, Jonathan R Shewchuk, and James F O'Brien. 2013. Simulating liquids and solid-liquid interactions with Lagrangian meshes. *ACM Transactions on Graphics (TOG)* 32, 2 (2013).

Blender Online Community. 2022. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam. http://www.blender.org

Peter Constantin and Gautam Iyer. 2007. A stochastic Lagrangian representation of the 3-dimensional incompressible Navier-Stokes equations. *Communications on Pure and Applied Mathematics* 61 (2007).

Robert L Cook. 1986. Stochastic sampling in computer graphics. *ACM Transactions on Graphics (TOG)* 5, 1 (1986).

Georges-Henri Cottet and Petros D Koumoutsakos. 2000. *Vortex methods: theory and practice*. Vol. 8. Cambridge university press Cambridge.

Benoit Couët, Oscar Buneman, and Anthony Leonard. 1981. Simulation of three-dimensional incompressible flows with a vortex-in-cell method. *J. Comput. Phys.* 39, 2 (1981), 305–328. https://doi.org/10.1016/0021-9991(81)90154-6

Ana Bela Cruzeiro. 2020. Stochastic Approaches to Deterministic Fluid Dynamics: A Selective Review. *Water* 12, 3 (2020). https://doi.org/10.3390/w12030864

Freddy Delbaen, Jinniao Qiu, and Shanjian Tang. 2015. Forward-Backward Stochastic Differential Systems Associated to Navier-Stokes Equations in the Whole Space. *Stochastic Processes and their Applications* 125 (2015).

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation'96*. Springer.

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics (TOG)* 26, 1 (2007).

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*.

Nicholas I. Fisher. 1995. *Statistical Analysis of Circular Data*. Cambridge University Press. https://doi.org/10.1017/CBO9780511564345

Nick Foster and Ronald Fedkiw. 2001. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*.

Cindy M Goral, Kenneth E Torrance, Donald P Greenberg, and Bennett Battaile. 1984. Modeling the interaction of light between diffuse surfaces. *ACM SIGGRAPH computer graphics* 18, 3 (1984), 213–222.

L Greengard and V Rokhlin. 1987. A fast algorithm for particle simulations. *J. Comput. Phys.* 73, 2 (1987), 325–348. https://doi.org/10.1016/0021-9991(87)90140-9

Toshiya Hachisuka. 2005. Combined Lagrangian-Eulerian approach for accurate advection. In *ACM SIGGRAPH 2005 Posters*. 114–es.

David AB Hyde and Ronald Fedkiw. 2019. A unified approach to monolithic solid-fluid coupling of sub-grid and more resolved solids. *J. Comput. Phys.* 390 (2019), 490–526.
Y. Le Jan and A. S. Sznitman. 1997. Stochastic cascades and 3-dimensional Navier–Stokes equations. *Probability Theory and Related Fields* 109 (1997).
Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. 2008. Radiance caching for participating media. *ACM Transactions on Graphics (TOG)* 27, 1 (2008), 1–11.
Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015).
James T Kajiya. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*.
Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A Machine Learning Approach for Filtering Monte Carlo Noise. *ACM Trans. Graph.* 34, 4, Article 122 (jul 2015), 12 pages. https://doi.org/10.1145/2766977
Jaroslav Krivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. 2005. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics* 11, 5 (2005), 550–561.
Wei Li, Kai Bai, and Xiaopei Liu. 2018. Continuous-scale kinetic fluid simulation. *IEEE Transactions on Visualization and Computer Graphics* 25, 9 (2018), 2694–2709.
Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling. *ACM Trans. Graph.* 39, 4, Article 47 (jul 2020), 20 pages. https://doi.org/10.1145/3386569.3392400
Thomas Lundgren and Petros Koumoutsakos. 1999. On the generation of vorticity at a free surface. *Journal of Fluid Mechanics* 382 (1999), 351–366.
Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2021. Fast and versatile fluid-solid coupling for turbulent flow simulation. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–18.
Sylvain Maire and Etienne Tanre. 2013. Monte Carlo approximations of the Neumann problem. *Walter de Gruyter GmbH* 19 (2013).
Gisiro Maruyama. 1955. Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo* 4, 1 (1955), 48–90.
Olivier Mercier, Xi-Yuan Yin, and Jean-Christophe Nave. 2013. The characteristic mapping method for the linear advection of arbitrary sets. *arXiv preprint arXiv:1309.2731* (2013).
Nicholas Metropolis and Stanislaw Ulam. 1949. The monte carlo method. *Journal of the American statistical association* 44, 247 (1949).
Zackary Misso, Benedikt Bitterli, Iliyan Georgiev, and Wojciech Jarosz. 2022. Unbiased and consistent rendering using biased estimators. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.
Marek Krzysztof Misztal, Kenny Erleben, Adam Bargteil, Jens Fursund, Brian Bunch Christensen, Jakob Andreas Bærentzen, and Robert Bridson. 2013. Multiphase flow of immiscible fluids on unstructured moving meshes. *IEEE transactions on visualization and computer graphics* 20, 1 (2013).
Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. 2009. Energy-Preserving Integrators for Fluid Animation. *ACM Trans. Graph* 28 (2009). https://doi.org/10.1145/1531326.1531344
Matthias Müller, David Charypar, and Markus H Gross. 2003. Particle-based fluid simulation for interactive applications.. In *Symposium on Computer animation*.
Mervin E Muller. 1956. Some continuous Monte Carlo methods for the Dirichlet problem. *Annals of Mathematical Statistics* 27, 3 (1956).
Mohammad Sina Nabizadeh, Albert Chern, and Ravi Ramamoorthi. 2021. Kelvin Transformations for Simulations on Infinite Domains. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).
Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo methods for volumetric light transport simulation. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library.
Bernt Oksendal. 2013. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media.
Sang Il Park and Myoung Jun Kim. 2005. Vortex fluid for gaseous phenomena. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*.
MF Peeters, WG Habashi, and EG Dueck. 1987. Finite element stream function-vorticity solutions of the incompressible Navier-Stokes equations. *International journal for numerical methods in fluids* 7, 1 (1987).
Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics (TOG)* 31, 4 (2012).
Matt Pharr. 2018. Guest Editor's Introduction: Special Issue on Production Rendering. *ACM Transactions on Graphics (TOG)* 28, 3 (2018). https://doi.org/10.1145/3212511
Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2018. *Physically based rendering: From theory to implementation*. Morgan Kaufmann.
Yang Qi, Dario Seyb, Benedikt Bitterli, and Wojciech Jarosz. 2022. A bidirectional formulation for Walk on Spheres. *Computer Graphics Forum (Proceedings of EGSR)* 41, 4 (July 2022). https://doi.org/10.1111/cgf.14586
Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Transactions on Graphics (TOG)* 38, 4 (2019).
Christian P. Robert and George Casella. 1999. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg.
Takahiro Sato, Christopher Batty, Takeo Igarashi, and Ryoichi Ando. 2018. Spatially adaptive long-term semi-Lagrangian method for accurate velocity advection. *Computational Visual Media* 4, 3 (2018).
Rohan Sawhney and Keenan Crane. 2020. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Transactions on Graphics (TOG)* 39, 4 (2020).
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).
Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. 2005. A vortex particle method for smoke, water and explosions. In *ACM SIGGRAPH*.
Naoharu H. Shimada and Toshiya Hachisuka. 2020. Quantum Coin Method for Numerical Integration. *cgforum* 39 (2020).
Nikolai A Simonov. 2017. Walk-on-spheres algorithm for solving third boundary value problem. *Applied Mathematics Letters* 64 (2017), 156–161.
Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*.
Geoffrey Ingram Taylor and Albert Edward Green. 1937. Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences* 158, 895 (1937), 499–521.
Jerry Tessendorf and Brandon Pelfrey. 2011. The characteristic map for fast and efficient vfx fluid simulations. In *Computer Graphics International Workshop on VFX, Computer Animation, and Stereo Movies. Ottawa, Canada*.
Eric Veach and Leonidas J Guibas. 1997. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 65–76.
Jiří Vorba and Jaroslav Křivánek. 2016. Adjoint-driven Russian Roulette and Splitting in Light Transport Simulation. *ACM Transactions on Graphics (TOG)* 35, 4, Article 42 (2016). https://doi.org/10.1145/2897824.2925912
Gregory J Ward, Francis M Rubinstein, and Robert D Clear. 1988. A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. 85–92.
Shiying Xiong, Rui Tao, Yaorui Zhang, Fan Feng, and Bo Zhu. 2021. Incompressible Flow Simulation of Vortex Segment Clouds. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).
Dongbin Xiu and George Em Karniadakis. 2001. A semi-Lagrangian high-order method for Navier–Stokes equations. *Journal of computational physics* 172, 2 (2001), 658–684.
Shuqi Yang, Shiying Xiong, Yaorui Zhang, Fan Feng, Jinyuan Liu, and Bo Zhu. 2021. Clebsch Gauge Fluid. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).
Xinxin Zhang and Robert Bridson. 2014. A PPPM fast summation method for fluids and beyond. *ACM Transactions on Graphics (TOG)* 33, 6 (2014).
Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005).