

Mixwell: Sharp 2D Fluid Brushes for Progressive Physics-Based Mixing

DOUG L. JAMES, Stanford University, USA
ETHAN JAMES, Independent Researcher, USA



Fig. 1. **Progressive Simulation and Rendering of 2D Fluid Mixing:** Our family of 2D Mixwell brushes enables fast evaluation of reverse-drift flows for cylindrical tines moving through ideal fluids. They support finite and infinite strokes in real time, and can be computed one sample at a time, enabling arbitrary resolution and progressive simulation and rendering. Shown are marbled scarf patterns generated on-the-fly in a Houdini Redshift OSL shader: (Left) Bouquet (Peacock), (Middle) Nonpareil-GelGit, and (Right) Feather. Please see our supplemental material for a 16K path-traced image. (3D models by Istinnaya.)

We present Mixwell, a family of sharp 2D fluid brushes and GPU-accelerated analytical methods for progressive, resolution-independent physics-based mixing. Derived from idealized potential flow around a cylindrical tine, Mixwell includes a cusped, Kelvinlet-style regularized velocity brush that parsimoniously captures cylinder–fluid interactions. Building on Maxwell’s 1869 drift formulation, we develop GPU-friendly evaluation strategies for particle drift in both infinite and finite strokes. For image-based workflows, we introduce cylindrical Reverse-Drift Functions (RDFs), displacement fields that encode tine insertion, motion, and removal. Like signed distance fields in geometric modeling, RDFs compose naturally and can be chained in shaders to model complex operations while avoiding intermediate texture blur. We also propose a periodicity-exploiting composition scheme for complex RDF patterns. All Mixwell operations are evaluated independently per sample, enabling truly progressive mixing and rendering without global solves, grids, or intermediate texture resampling. We demonstrate real-time GLSL and HLSL implementations and production integrations in Houdini (OpenCL, OSL), enabling progressive, arbitrary-resolution mixing and rendering with negligible numerical dissipation.

Authors’ Contact Information: Doug L. James, Stanford University, Stanford, California, USA, djames@cs.stanford.edu; Ethan James, Independent Researcher, Half Moon Bay, California, USA, sparki262@gmail.com.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 1557-7368/2026/7-ART79
<https://doi.org/10.1145/3811312>

ACM Reference Format:

Doug L. James and Ethan James. 2026. Mixwell: Sharp 2D Fluid Brushes for Progressive Physics-Based Mixing. *ACM Trans. Graph.* 45, 4, Article 79 (July 2026), 18 pages. <https://doi.org/10.1145/3811312>

1 Introduction

Fluidic sculpting and mixing tools are visually striking elements of modern modeling and animation workflows. Ad hoc geometric deformers can produce fluid-like effects inexpensively, but they often lack physical realism and incompressibility control. Physics-based fluid simulation can offer high realism and incompressibility, but it is computationally expensive, resolution-limited, and prone to numerical dissipation and blur. Analytical deformation brushes, such as Kelvinlets [de Goes and James 2017], provide elegant closed-form flow fields, but they do not precisely model fluid-brush mixing interactions, and long strokes still require fine time stepping. In short, we lack a sharp, controllable, incompressible 2D fluid brush that captures precise physical, tine-like flow interactions and supports detailed arbitrary-resolution fluid effects with negligible dissipation and blur at real-time rates.

This paper introduces Mixwell, a special-purpose family of sharp 2D fluid brushes derived directly from idealized potential flow around an inserted cylindrical tine. Directly inspired by Maxwell’s 1869 drift formulation for 2D flow around a cylinder [Maxwell 1869], we develop GPU-friendly analytical and numerical methods to evaluate particle drift for both infinite and finite strokes. We first leverage

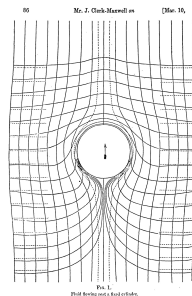
classical analytical techniques for 1D drift to efficiently simulate high-resolution long brush strokes; then, we analyze analytical 2D drift calculations and overcome mathematical and physical limitations. In the process, we derive a novel cusped regularized Kelvinlet-style, velocity-level brush on $\mathbb{R}^2$, our Mixwell brush, that parsimoniously encodes cylinder-fluid interactions previously studied by Maxwell and can be applied directly and efficiently to geometry or texture coordinates. This yields our position-level Mixwell brush and advection scheme, which advances particles under the influence of short-or-long, straight-or-curved brush strokes using a combination of analytical and numerical methods.

To enable practical image-based modeling workflows, we introduce Reverse-Drift Functions (RDFs), which encode stroke-induced texture displacements, including time insertion, motion, and removal. Analogous to Signed Distance Fields (SDFs) for geometric modeling, RDFs can be composed to model complex fluid mixing operations such as combing, raking, and wave-like brush interactions. RDFs can be reverse-chained directly in shaders, avoiding intermediate texture sampling and blurring. For complex patterns, we further introduce a periodicity-exploiting RDF composition scheme that enables efficient evaluation of repeated and layered stroke structures. We demonstrate real-time graphics implementations in GLSL and HLSL, as well as production implementations in Houdini Copernicus (OpenCL) and RedShift (OSL, see Figure 1), producing high-resolution imagery with negligible numerical dissipation.

Contributions. This paper contributes: (i) a sharp, cylinder-derived incompressible 2D velocity brush suitable for Kelvinlet-style interaction; (ii) position-level advection schemes for finite and infinite strokes based on drift coordinates; (iii) GPU-efficient evaluation strategies enabling progressive simulation and rendering of fluid brush effects; (iv) stroke-based Reverse-Drift Functions (RDFs) with insertion and removal, supporting composable, blur-free image-space mixing; (v) a periodicity-exploiting RDF composition scheme for efficient modeling of complex, repeated mixing patterns.

2 Related Work

Marbling and Drift. Marbling is an ancient art practiced across many cultures, including Suminagashi in Japan [Chambers 1991] and Ebru in Turkey and Persia, with a well-documented history (see surveys by Brett [2021]; Wolfe [1990]). The Maxwell connection is mathematical and direct. In 1869, Maxwell [1869] modeled the 2D displacement of a fluid particle due to a passing cylinder using elliptic integrals and explicitly suggested the induced drift as a model for paper marbling [Maxwell 1890]. He even rendered a marbled grid by hand, noting that “If a maker of ‘marbled’ paper were to rule the surface of his bath with straight lines of paint at right angles, and then to draw a cylindrical ruler through the bath up to the middle, and apply the painted lines to his paper, he would produce the design” shown inset. Later, Darwin [1953] independently advanced these net particle drift calculations, making them widely known [Milne-Thomson 1996], and the phenomenon is often discussed as Darwin (or Darwinian) drift, despite Maxwell’s earlier cylinder



analysis. We revisit classical analytical drift techniques, adapt them for GPU simulation, and introduce a novel brush for “Mixwell flow” modeling that enables fast and robust GPU drift simulation without elliptic integrals.

For the classical drift model, we model the surrounding flow using classical potential flow, i.e., the incompressible, irrotational motion of an inviscid fluid past an infinitely long cylinder. While the velocity field and streamlines are readily available, computing particle travel times and the resulting net displacement in closed form leads to elliptic integrals and semi-implicit relations [Milne-Thomson 1996; Shōji et al. 2009]. We evaluate these integrals using Carlson’s symmetric integral forms [Carlson 1995] and a custom Newton solver, both well suited for GPU evaluation, and we compare them to our Mixwell brush advection scheme.

Deformation Brushes. Interactive editing, as the composition of local deformation tools, has a long history in graphics. Early swept-tool deformers include Sweepers and Swirling-Sweepers [Angelidis et al. 2006; Funck et al. 2006]. Regularized Kelvinlets [de Goes and James 2017, 2019] provide closed-form elastic deformation brushes derived as fundamental solutions of linear elasticity and incompressible fluid flow (Stokeslets [Cortez 2001]) for regularized force loads. These physics-based brushes smoothly drag a local neighborhood along with the flow using a blob of force, but they do not specifically model the flow around a cylinder: there is no circular exclusion domain that material flows around at different rates, inducing singular drift. Surprisingly, the closest Kelvinlet approximation to our Mixwell brush is the 2D cusped Laplacian Kelvinlet (cf. Appendix A.2 of [de Goes and James 2019]), which has an ad hoc construction but exhibits a similar asymptotic cusp profile and far-field decay rate after suitable parameter fitting. However, Kelvinlets are primarily used as displacement-based deformers for sculpting and space deformation. Haetinger and Gastal [2020] investigated real-time image deformation and video time warping using Newton-based inversion of regularized Kelvinlet brushes, which work best for small and smooth deformations but can incur image blur during sampling. Such approaches are less appealing for large and/or sharp deformations, or multi-brush interactions in complex mixing processes, all of which are difficult to invert numerically. Mixwell targets a complementary regime: sharp, incompressible 2D fluidic brushes whose velocity fields are specifically derived from ideal cylindrical flow, which can be used to construct cylindrical RDFs for texture advection using reverse-time advection solvers, and support arbitrarily long brush strokes without timestep-induced image blur. We rely on reverse-time-integrated cylindrical RDFs (and other techniques) to perform large-time texture advection and reduce image blur.

Progressive quasistatic evaluation has been explored for cloth and shells [Zhang et al. 2022, 2023] and for fluids via adaptive grids and multiscale or reduced-order enrichments [Kim et al. 2008b; Losasso et al. 2004; Treuille et al. 2006; Wicke et al. 2009]. In contrast, Mixwell’s 2D drift model admits accurate, pointwise, resolution-independent evaluation, allowing fluid deformation and rendering to be computed together progressively.

Analytical Mixing Flows. Aref [1984] pioneered chaotic advection using time-periodic flows as finite-time integrable, area-preserving

maps. Along with Baker’s transformation and Horseshoe maps [Ottino 1989], such analytically tractable flows enable forward particle tracking to quantify mixing. Mixwell similarly employs flow “brushes” (point vortices in Aref; cylinders here), but instead of forward mixing analysis, we develop RDFs to enable backward texture sampling for rendering.

Digital Marbling Systems. There is a long history of digital marbling in graphics, but no analytical, physics-based incompressible mixing brush has emerged that is consistent with cylinder flow. Geological marble patterns are commonly approximated using procedural noise [Lagae et al. 2010; Perlin 1985]. For 2D hydrographic marbling, the direct numerical simulation of the 2D Navier-Stokes equations has been explored for two decades: AtelierM [Mao et al. 2003] and Akgun [2004] pioneered interactive marbling using Stable Fluids [Stam 1999]. However, these early systems suffered from numerical dissipation, blurring, and limited resolution. Subsequent work explored GPU-accelerated Poisson solvers [Gazziro et al. 2018; Jin et al. 2007] and better advection schemes—CIP [Yabe et al. 2001], BFECC [Kim et al. 2007], MacCormack [Selle et al. 2008]—culminating in AtelierM++ [Zhao et al. 2009] with GPU-accelerated USCIP [Kim et al. 2008a] and edge-enhancing filters to reduce blur. Vector graphics approaches [Ando and Tsuruno 2011; Liu and Chen 2012; Weiskopf 2004] preserve sharp contours but remain limited by their underlying coarse fluid grids.

The mathematical marbling work of Lu et al. [2012] proposed closed-form “tool functions” under incompressible, immiscible 2D ink assumptions, emphasizing sharp contours, tractable inverses, and GPU backward mapping for real-time feedback [Lu et al. 2013]. Their radial ink-drop insertion model is based on an equal-area argument and is similar to our time insertion/removal maps. The paper also introduces a sharp, stylus-like brush that does not satisfy the incompressible flow assumption for finite strokes and is (in the terminology of [de Goes and James 2019]) a “grab brush.” In contrast, the Mixwell brush is derived as the physical pullback of the ideal cylinder flow’s velocity field through the area-preserving time insertion/removal map, yielding incompressible Mixwell flow (see comparison in §7.3). We derive both a sharp Kelvinlet-style fluid brush and advection schemes for finite, incompressible brush strokes, supporting image deformations of arbitrary resolution.

Although not strictly marbling, computational hydrographic printing techniques simulate and correct for texture distortion when dipping objects into a water bath with an inkjet-printed PVA film layer modeled as an elastic membrane, enabling precise alignment of surface textures on complex 3D surfaces [Zhang et al. 2015].

Image Advection and Blur Avoidance. Blurring from repeated resampling is a classic failure mode in image and volume advection. Sims [1992] emphasized composing deformations and sampling once rather than repeatedly deforming intermediate images. In fluid simulation, semi-Lagrangian advection [Stam 1999] and texture-coordinate advection variants [Kim et al. 2008b] reduce diffusion but still face resampling artifacts. Tessendorf and Kowalski [2011] introduced Semi-Lagrangian Mapping (SELMA) for “resolution independent volumes.” Mixwell’s Reverse-Drift Function (RDF) composition plays a similar role for high-frequency mixing: composing

analytical displacement fields into a single texture lookup (with supersampling where needed) rather than accumulating blur through many small advection steps.

Characteristic-Map and Flow-Map Methods. These strategies belong to a class of methods for representing fluid transport as a composable flow map, rooted in the classical method of characteristics and applied to 2D transient fluid flow as early as Wiggert and Wylie [1976]. Tessendorf and Pelfrey [2011] later brought the term *characteristic map* to graphics, framing it as a generalization of gridless advection and the SELMA scheme above. In computational physics, the Characteristic Mapping Method (CMM) evolves a long-time flow map as a composition of accurate short-time submaps, achieving high-order resolution on coarse grids for linear advection [Mercier et al. 2020] and the 2D incompressible Euler equations [Yin et al. 2021]. In computer graphics, many recent works have explored this perspective: BiMocq² evolves bidirectional mappings on grids [Qu et al. 2019]; covector and impulse formulations advect velocity-like covector quantities along the map to preserve circulation [Nabizadeh et al. 2022]; Neural Flow Maps encode the spatiotemporal velocity field implicitly for long-range bidirectional advection [Deng et al. 2023]; Particle Flow Maps assemble accurate long-range maps from Lagrangian particle trajectories [Zhou et al. 2024], and Eulerian vortex methods advect vorticity along flow maps [Wang et al. 2024]. In contrast, Mixwell RDFs are built from (semi-)closed-form analytical drift maps for potential flow around a cylinder, rather than numerically integrating maps from a simulated velocity field. Mixwell is therefore not a Navier-Stokes solver and is restricted to the specialized 2D quasi-static brush setting we target, but it admits pointwise, resolution-independent map evaluation with no per-step integration error in the infinite-stroke case and supports fast GPU shader evaluation.

3 Idealized Cylinder Flow

3.1 2D Potential Flow Around a Cylinder

Following [Maxwell 1869; Shōji et al. 2009], we model the flow as classical 2D potential flow of an incompressible, irrotational, inviscid fluid past an infinitely long cylinder. Let the cylinder have a radius ε and be fixed at the origin in the body frame (X, Y) . In this frame, the far-field flow is uniform with unit speed in the negative X direction, and a tracer particle at $P = (X, Y)$ will follow the stream function,

$$\Psi(P) = \left(1 - \frac{\varepsilon^2}{X^2 + Y^2}\right) Y, \quad (1)$$

such that if it starts at $P_0 = (X_0, Y_0)$ then all trajectory points satisfy $\Psi(P) = \Psi(P_0) = \eta_\infty$ where $\eta_\infty = \lim_{t \rightarrow \pm\infty} Y(t)$ is the signed height value of the streamline as $|X| \rightarrow \infty$. The associated body-frame velocity field is

$$\mathbf{V}_B(P) = \left(-\frac{\partial \Psi}{\partial Y}, \frac{\partial \Psi}{\partial X}\right) = \left(-1 + \frac{\varepsilon^2(X^2 - Y^2)}{(X^2 + Y^2)^2}, \frac{2\varepsilon^2 XY}{(X^2 + Y^2)^2}\right) \quad (2)$$

so that the equations of motion of the tracer particles around the cylinder are $\dot{P} = (\dot{X}, \dot{Y}) = \mathbf{V}_B(P)$ for $P \in \Omega_\varepsilon$ where $\Omega_\varepsilon = \mathbb{R}^2 \setminus \bar{B}_\varepsilon$ and $\bar{B}_\varepsilon = \{x \in \mathbb{R}^2 : \|x\| \leq \varepsilon\}$.

Finally, when the cylinder is at the origin, the instantaneous world-frame spatial velocity field (which is zero at infinity) is

$$\mathbf{V}_W(\mathbf{P}) = \mathbf{V}_B(\mathbf{P}) + (1, 0). \quad (3)$$

These streamlines and velocity fields are shown in Figure 9(Left).

3.2 Maxwell drift and time along streamlines

3.2.1 Time-Stepping Particle Drift. The deflection of a fluid particle's position by the passing of a moving cylinder is referred to as "drift." The brute-force method to compute it by time stepping an ODE is as follows. Let the body-frame position of the particle be $\mathbf{P} = \mathbf{P}_B = (X, Y) \in \Omega_\varepsilon$ (omitting "B" subscripts), and the world-frame position be $\mathbf{P}_W = (X_W, Y_W) \in \Omega_{\varepsilon, W}(t)$ where $\Omega_{\varepsilon, W}(t)$ is the punctured domain in which the cylinder is moving. Without loss of generality, assume a unit cylinder radius ($\varepsilon = 1$), that the frames differ only by a unit X velocity shift, and the two frames coincide at time $t = 0$, so that

$$\mathbf{V}_W = \mathbf{V}_B + (1, 0) \iff \dot{X}_W = \dot{X} + 1, \dot{Y}_W = \dot{Y} \quad (4)$$

$$\mathbf{P}_W = \mathbf{P}_B + (t, 0) \iff X_W = X + t, Y_W = Y. \quad (5)$$

The world-frame motion of a drifting particle is governed by the differential equation,

$$\dot{\mathbf{P}}_W = \mathbf{V}_W = \mathbf{V}_B(X_W - t, Y_W) + (1, 0), \quad (6)$$

which, substituting $\mathbf{V}_B(X, Y)$ from (2), yields an ODE for particle motion in the punctured domain:

$$\dot{X}_W = \frac{(X_W - t)^2 - Y_W^2}{((X_W - t)^2 + Y_W^2)^2}, \quad \dot{Y}_W = \frac{2(X_W - t)Y_W}{((X_W - t)^2 + Y_W^2)^2}. \quad (7)$$

Note that particles passing close to the cylinder must be time-stepped carefully to prevent them from crossing into the cylinder and becoming trapped.

Initial Conditions. At time $t = 0$, let the test particle be located on the Y axis, $X_W(0) = X(0) = 0$, with a height of $Y_W(0) = Y(0) = \eta_0$. This zero-crossing height η_0 is related to η_∞ by the stream function (1),

$$\eta_\infty = \Psi(X, Y) = \left(1 - \frac{1}{X^2 + Y^2}\right) Y \quad (8)$$

and $\Psi(0, \eta_0) = \eta_\infty$, so $\eta_\infty = \eta_0 - 1/\eta_0$. Solving for η_0 , we obtain

$$\eta_0 = \frac{\eta_\infty}{2} \left(1 + \sqrt{1 + \frac{4}{\eta_\infty^2}}\right) \stackrel{\eta_\infty > 0}{=} \frac{1}{2} \left(\eta_\infty + \sqrt{\eta_\infty^2 + 4}\right). \quad (9)$$

Note that $|\eta_0| \in (1, \infty)$ if the limit height $|\eta_\infty| \in (0, \infty)$. In summary, given any point (X, Y) , we can find the streamline's characteristic values, η_∞ then η_0 .

Drift Initial Value Problem (IVP). These initial conditions, $\mathbf{P}_W(0) = (0, \eta_0)$, at $t = 0$, combined with the ODE in (7) constitute the canonical drift initial value problem (IVP) that can be time stepped to obtain $\mathbf{P}_W(t) = (X_W(t), Y_W(t))$. While it provides a brute-force ground truth estimator, the direct numerical simulation of particle advection in the punctured domain $\Omega_{\varepsilon, W}(t)$ can be expensive and a source of approximation errors for efficient brush simulation. For example, the ubiquitous scenario of computing the X_W deflection after a cylinder passes by (from $X_W = \infty$ to $-\infty$)—something computed millions or billions of times in typical scenarios—is equivalent to

time stepping the IVP to get $X_W(-\infty) - X_W(+\infty) = 2X_W(-\infty)$, which is a long-time integration problem with singularities and domain exclusions, and is not something we want to compute a billion times per frame. Instead, we seek a closed-form approximation and/or a fast evaluation strategy.

3.2.2 Classical Analytical Solution for Drift. Analytical solutions for particle drift are classical results and are reported below; see Appendix A for standard derivations [Darwin 1953; Maxwell 1869; Shōji et al. 2009].

Analytical solutions to the drift IVP exist but are implicit in time:

$$X_W = X_W(\sigma, \eta_\infty) = \int_0^\sigma \frac{\cos 2\theta}{\sqrt{\eta_\infty^2 + 4 \cos^2 \theta}} d\theta, \quad (10)$$

$$Y_W = Y_W(\sigma, \eta_\infty) = \frac{1}{2} \left(\eta_\infty + \sqrt{\eta_\infty^2 + 4 \cos^2 \sigma} \right), \quad (11)$$

$$t = t(\sigma, \eta_\infty) = -\frac{1}{4} \int_0^\sigma \frac{(\eta_\infty + \sqrt{\eta_\infty^2 + 4 \cos^2 \theta})^2}{\cos^2 \theta \sqrt{\eta_\infty^2 + 4 \cos^2 \theta}} d\theta, \quad (12)$$

where $t \in (-\infty, 0]$ and $\sigma \in [0, \pi/2]$ for (X, Y) in the upper quadrant. Unfortunately, instead of providing the trajectory as $\mathbf{P}_W(t, \eta_\infty)$, it is given by a system of integral equations that are implicit in time and expressed in terms of the angle

$$\sigma = \text{sgn}(X) \arccos \frac{Y}{R}. \quad (13)$$

where $R = \sqrt{X^2 + Y^2}$. This (σ, η_∞) coordinate system is visualized in Figure 3. We refer to $t(\sigma, \eta_\infty)$ in (12) as *Maxwell's time integral*. These integrals can be evaluated numerically to plot the trajectory of the particle $\mathbf{P}_W(t)$ as the cylinder passes [Shōji et al. 2009], analogous to how we can directly timestep the ODEs in (7).

3.3 2D Drift for Infinite and Finite Strokes

We now consider important 2D drift calculations for straight brush strokes that are either very long (infinite) or of finite length.

3.3.1 Infinite Strokes. An important special case for fluid mixing are long line-like strokes, such as those that occur in popular Nonpareil and Gel-Git marbling patterns. Given the time trajectory, we can transform the particle's position into the canonical unit-radius body frame and determine the η_∞ of the trajectory, which also represents the beginning and end $Y_W = Y$ coordinates of the trajectory. It only remains to find the X_W drift as the comb passes by, which (by symmetry) is *twice* the value of the X_W integral (10) as $t \rightarrow -\infty$ ($\sigma \rightarrow \pi/2$):

$$\xi(\eta_\infty) \equiv 2 \int_0^{\pi/2} \frac{\cos 2\theta}{\sqrt{\eta_\infty^2 + 4 \cos^2 \theta}} d\theta, \quad (14)$$

$$= \frac{-1}{|\eta_\infty|} \left((\eta_\infty^2 + 2) K \left[\frac{-4}{\eta_\infty^2} \right] - \eta_\infty^2 E \left[\frac{-4}{\eta_\infty^2} \right] \right) \quad (15)$$

which is a negative cusp-like displacement valid for $|\eta_\infty| > 0$ plotted in Figure 4; although written differently, (15) is mathematically consistent with [Darwin 1953; Maxwell 1869; Milne-Thomson 1996; Shōji et al. 2009].

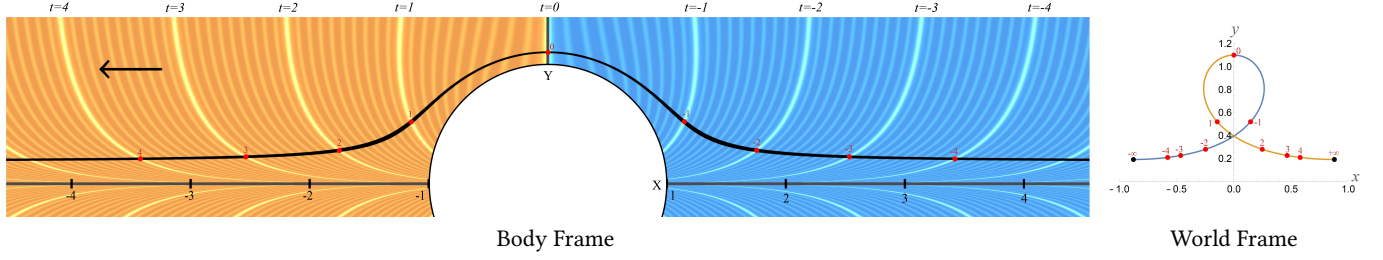


Fig. 2. **Maxwell Time and Drift Curves** for a unit cylinder moving to the right at unit velocity, $V = (1, 0)$. (Left) Body frame depiction of leftward flow around the cylinder, with Maxwell time plotted as time t to flow from the vertical Y axis ($t = 0$) to a point, with upwind values in blue ($t < 0$), and downwind values shown in orange ($t > 0$). A streamline corresponding to $\eta_0 = 1.1$ ($\eta_\infty = 21/110$) is plotted, with Maxwell time values (in red) along the curve. (Right) World frame particle trajectory has the same $Y_W = Y$ values but different $X_W = X - t$ values, and shows a characteristic drift curve as the particle moves around the passing cylinder, and gets dragged to the right (integer times shown dotted in red). Relative 2D drift displacements during advection correspond to differences between these points along the curve. Importantly, the particle's 1D reverse drift, $\xi < 0$, is the difference between the end and start X_W positions, $\xi(\eta_\infty) = X_W(-\infty) - X_W(\infty)$.

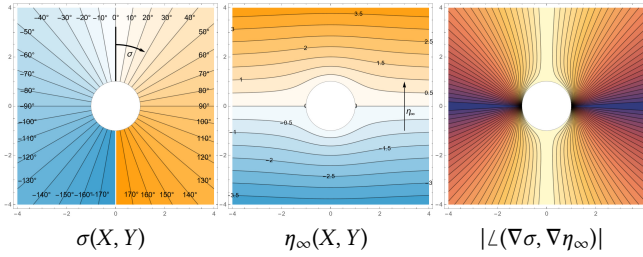


Fig. 3. **Coordinate systems and conditioning:** Visualization of (Left) $\sigma \in (-\pi, \pi]$ measuring the angle from the vertical Y axis, (Middle) η_∞ contours, and (Right) the $|\angle|$ between the coordinates' gradients showing the poor conditioning of this coordinate system along the X axis, which unfortunately also corresponds to the singularity in Maxwell time, $t(X, Y)$.

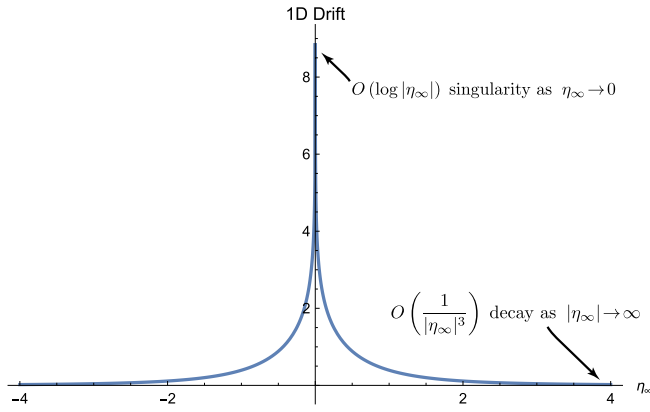


Fig. 4. **Plot of 1D drift magnitude**, $-\xi(\eta_\infty) > 0$, corresponding to an infinite stroke of a single circular tine. Features include the logarithmic singularity as $\eta_\infty \rightarrow 0$ that streaks paint along the tine's path, and the fast far-field decay as $|\eta_\infty| \rightarrow \infty$.

Here $K[m]$ and $E[m]$ are complete elliptic integrals of the first and second kinds, respectively, with parameter $m < 1$ (see Figure 5):

$$K[m] \equiv \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - m \sin^2 \theta}}, \quad E[m] \equiv \int_0^{\pi/2} \sqrt{1 - m \sin^2 \theta} d\theta. \quad (16)$$

Since $|\eta_\infty| > 0$ we must evaluate K and E for $m = -4/\eta_\infty^2 \in (-\infty, 0)$. The reverse drift $\xi(\eta_\infty)$ is the key quantity to calculate for the nonpareil marbling and depends only on the distance from the path of the tine, $Y = |\eta_\infty|$ in the canonical unit-radius coordinates. As a reference for implementers, the drift is $\xi \approx -0.393175$ at $\eta_\infty = 1$.

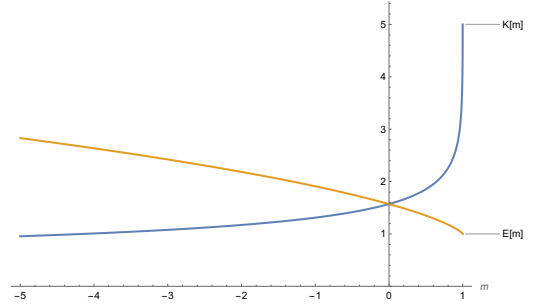


Fig. 5. **Infinite strokes use complete elliptic integrals** of the first kind, $K[m]$ and the second kind, $E[m]$, with the parameter $m = -4/\eta_\infty^2 \in (-\infty, 0)$.

3.3.2 Fast Series Approximation using Matched Asymptotic Expansions. Given the analytical formula for drift, ξ , due to an infinite stroke at point-to-centerline distance η_∞ , we can evaluate it directly using GPU algorithms for elliptic functions (see supplemental material). However, given the numerous evaluations this function requires, we found it useful to numerically derive a C^0 matched asymptotic expansion in Mathematica using three ranges of $|\eta_\infty|$: (1) near zero, which has a logarithmic singularity; (2) a smooth mid-range model using a Padé expansion; and (3) the far-field expansion, which decays as $O(1/|\eta_\infty|^3)$. Detailed formulae are provided in Appendix B. This approximation is over an order of magnitude faster (e.g., 15x) than directly evaluating the elliptic integrals using fast Carlson iterations in our tests.

3.4 Finite Strokes

Evaluating integrals for X_W (10) in Mathematica and simplifying yields

$$X_W(\sigma, \eta_\infty) = \frac{(\eta_\infty^2 + 4) E(\sigma | m) - (\eta_\infty^2 + 2) F(\sigma | m)}{2\sqrt{\eta_\infty^2 + 4}}, \quad (17)$$

with elliptic parameter $m = \frac{-4}{\eta_\infty^2 + 4} \in (0, 1)$. The main costs of evaluating X_W are $F(\sigma|m)$ and $E(\sigma|m)$, which are incomplete elliptic integrals of the first and second kinds, respectively, with parameter m :

$$F(\phi|m) \equiv \int_0^\phi \frac{d\theta}{\sqrt{1 - m \sin^2 \theta}} \xrightarrow{\phi \rightarrow \pi/2} K[m], \quad (18)$$

$$E(\phi|m) \equiv \int_0^\phi \sqrt{1 - m \sin^2 \theta} d\theta \xrightarrow{\phi \rightarrow \pi/2} E[m]. \quad (19)$$

The trajectory time integral (12), $t = t(\sigma, \eta_\infty)$, can also be expressed in terms of elliptic integrals. However, using the definition $X_W = X + t$, we can write t in terms of X_W (assuming $\eta_\infty > 0$ here):

$$t(\sigma, \eta_\infty) = -X + X_W(\sigma, \eta_\infty) \quad (20)$$

$$X = Y \tan \sigma, \quad (21)$$

$$Y = \frac{1}{2} \left(\eta_\infty + \sqrt{\eta_\infty^2 + 4 \cos^2 \sigma} \right) > 0, \quad (22)$$

which can be verified from the definitions of η_∞ in (8) and σ in (54) to transform coordinates from (σ, η_∞) to (X, Y) . The key takeaway is that the two integrals for X_W (10) and t (12) can be expressed in terms of the same elliptic integrals for X_W in (17).

3.4.1 Discussion. Unlike the 1D reverse-drift formula for $\xi < 0$ in (15), which is due to an infinite stroke and its fast series expansion, the formulae for finite strokes' (X_W, Y_W, t) values cannot be immediately used to simulate mixing. This is partly because we need to perform advection to determine differences in (X_W, Y_W) values separated by time Δt (see Figure 6).

As part of our due diligence, we investigated leveraging the classical analytical drift expressions to build a one-step advection solver using a custom fast Newton iteration. While it can deliver accurate one-step solves for finite strokes, it is mathematically complicated, incurs significant GPU overhead due to special-function evaluations (XU), and, perhaps more importantly, suffers from coordinate and physical (Maxwell time) singularities that require patching. We therefore provide a detailed exploration and comparison of this Newton-based approach in the supplemental material; in brief, it is slower, more complex, and more prone to singularities and visual artifacts. We now turn to a much faster and simpler alternative based on our “Mixwell brush.”

4 The Sharp Mixwell Brush

Painters do not typically leave their brushes stuck in their paintings, and neither do we want to have circular exclusions in our fluid media that necessitate simulation on a punctured domain. Therefore, we mathematically consider each infinitesimal brush stroke to comprise the insertion, movement, and removal of a latent cylindrical brush. The resulting “Mixwell flow” describes a fluid velocity field on $\mathbb{R}^2$ that is the basis for our Mixwell brush (see demo in Figure 7).

4.1 Mixwell Flow Construction

Mathematically, we obtain the incompressible Sharp Mixwell velocity field by conjugating the canonical incompressible cylinder flow with an area-preserving radial insertion-removal map on the punctured plane, that is, by pulling back the cylinder velocity field under this incompressible coordinate transform [Lee 2013]. The

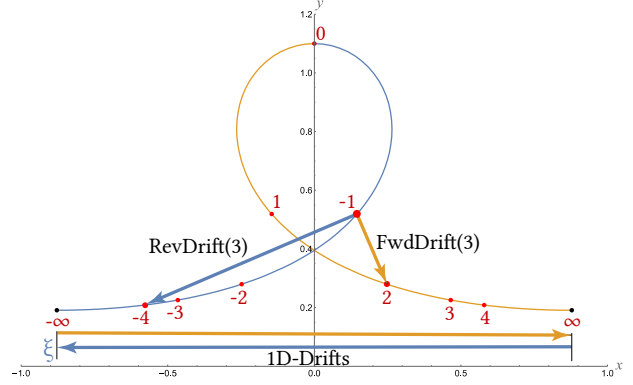


Fig. 6. **2D Drift Illustration:** The world-frame particle drift from Figure 2 annotated with (a) forward- and reverse-drift displacements of the point $p(-1)$ (at time $t = -1$) for $\Delta t = \pm 3$; and (b) horizontal 1D drift displacements shown at the bottom for the reverse-drift $\xi(\eta_\infty) < 0$ in blue, and the opposite 1D forward drift $(-\xi)$ in orange.



Fig. 7. **Mixwell Brush Deformation:** A bird-like figure marbled on a modest HD-resolution raster using a real-time Shadertoy implementation. The active stroke accumulates its deformation map separately to avoid blurring the total deformation map. These maps are stored at screen resolution and interpolated at lookup, so quality is below our fully supersampled results.

incompressible map used to model the fluid’s response due to the insertion ($\mathcal{I}_\epsilon(p)$) and removal ($\mathcal{I}_\epsilon^{-1}(P)$) of the radius- ϵ circular tine at the origin is described in Appendix C. Mathematically, these maps are used to pull back the cylinder velocity field $V_B(P)$ from Ω_ϵ to $\mathbb{R}^2$ as follows (see Figure 8).

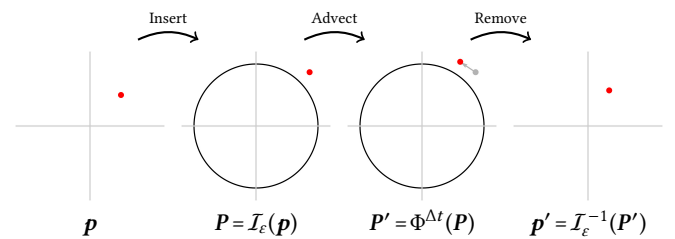


Fig. 8. **Mixwell Brush Construction**

Conceptually, the initial point $p \in \mathbb{R}^2$ is first expanded by the insertion of the tine to $P = \mathcal{I}_\epsilon(p)$. Second, the point is advected by V_B a small distance to $P' = P + V_B(P) \Delta t$. Finally, the tine is removed, contracting the point to its final position, $p' = \mathcal{I}_\epsilon^{-1}(P')$ or $p' = f(p, \Delta t) = \mathcal{I}_\epsilon^{-1}(\mathcal{I}_\epsilon(p) + V_B(\mathcal{I}_\epsilon(p)) \Delta t)$. Taking the limit as $\Delta t \rightarrow 0$, we can obtain the Mixwell brush velocity field,

$$v_B(p) = \lim_{\Delta t \rightarrow 0} \frac{f(p, \Delta t) - p}{\Delta t}. \quad (23)$$

Equivalently, $\mathbf{v}_B(\mathbf{p})$ is the pullback of the canonical incompressible cylinder flow $\mathbf{V}_B$ under the incompressible radial insertion map $\mathcal{I}_\varepsilon$,

$$\mathbf{v}_B(\mathbf{p}) = D\mathcal{I}_\varepsilon(\mathbf{p})^{-1} \mathbf{V}_B(\mathcal{I}_\varepsilon(\mathbf{p})), \quad (24)$$

where $D\mathcal{I}_\varepsilon(\mathbf{p})$ is the Jacobian of the insertion map (see Appendix C).

Analyzing in Mathematica, the resulting Sharp Mixwell velocity field (body frame) is

$$\mathbf{v}_B(\mathbf{p}) = \left(\frac{-r^4 - 2\varepsilon^2 r^2 + \varepsilon^2 x^2}{r r_\varepsilon^3}, \frac{\varepsilon^2 x y}{r r_\varepsilon^3} \right) \xrightarrow{r \rightarrow \infty} (-1, 0) \quad (25)$$

where $\mathbf{p} = [x, y] \in \mathbb{R}^2$, $r = \|\mathbf{p}\|$, and $r_\varepsilon = \sqrt{r^2 + \varepsilon^2}$. The instantaneous world-frame Mixwell velocity field is $\mathbf{v}_W(\mathbf{p}) = \mathbf{v}_B(\mathbf{p}) + (1, 0)$, which decays to 0 as $r \rightarrow \infty$. The insertion map also gives the Mixwell stream function, $\psi(\mathbf{p}) = \Psi(\mathcal{I}_\varepsilon(\mathbf{p})) = \frac{r}{r_\varepsilon} y$. These stream functions and velocity fields are plotted in Figure 9.

4.2 Kelvinlet brush form

We now report the 2D Sharp Mixwell brush solution $\mathbf{v}_W(\mathbf{p})$ using notation identical to regularized Kelvinlets [de Goes and James 2017, 2019]. 2D Kelvinlet brushes represent instantaneous world-space flow fields and can be expressed compactly in the standard 2-by-2 matrix form,

$$\mathcal{K}_\varepsilon(\mathbf{r}) = A_\varepsilon(r) \mathbf{I} + B_\varepsilon(r) \mathbf{r} \mathbf{r}^t, \quad (26)$$

for brush-specific functions, $A_\varepsilon(r)$ and $B_\varepsilon(r)$; here $\mathbf{r} = [x, y]$ is the position relative to the brush tip, $r = \|\mathbf{r}\|$, $r_\varepsilon = \sqrt{r^2 + \varepsilon^2}$, and ε is the radius of the cylindrical tine. Specifically, our 2D Sharp Mixwell functions can be identified as

$$A_\varepsilon(r) = 1 - \frac{r(r^2 + 2\varepsilon^2)}{r_\varepsilon^3}, \quad B_\varepsilon(r) = \frac{\varepsilon^2}{r r_\varepsilon^3}. \quad (27)$$

In this notation, the instantaneous world-frame Mixwell velocity field is $\mathbf{v}_W(\mathbf{p}) = \mathcal{K}_\varepsilon(\mathbf{p})(1, 0) = [\mathcal{K}_\varepsilon(\mathbf{p})]_{:1}$.

Kelvinlet brushes are typically used to evaluate “forward drift” in world space, e.g., to deform mesh vertices. For example, given a small brush tip displacement $\tilde{\mathbf{u}}$, a field point at a relative position $\mathbf{r}$ will observe forward-drift and reverse-drift displacements,

$$\mathbf{u}_{fd}(\mathbf{r}) = \mathcal{K}_\varepsilon(\mathbf{r}) \tilde{\mathbf{u}}, \quad \text{and} \quad \mathbf{u}_{rd}(\mathbf{r}) = -\mathcal{K}_\varepsilon(\mathbf{r}) \tilde{\mathbf{u}}, \quad (28)$$

respectively. At the tip ($\mathbf{r} = 0$) we recover $\mathbf{u}_{fd}(0) = \tilde{\mathbf{u}}$, since $\mathcal{K}_\varepsilon(0) = \mathbf{I}$. Finally, these are sharp 2D Mixwell brushes with an $O(1/r^2)$ far-field decay and a cusped profile, as we now show.

4.3 Sharp Brush Profile

Following de Goes and James [2019], we use the 2D Kelvinlet profile, defined as the $\mathcal{K}_{xx}$ response along the x axis,

$$P(x) \equiv \mathcal{K}_\varepsilon^{xx}([x, 0]) = A_\varepsilon(|x|) + B_\varepsilon(|x|)x^2 = 1 - \frac{|x|}{\sqrt{x^2 + \varepsilon^2}} \quad (29)$$

This scalar profile has a sharp cusp, $P(x) \approx 1 - \frac{|x|}{\varepsilon}$ for $|x| \ll \varepsilon$, and an inverse-squared far-field decay, $P(x) \approx \varepsilon^2/2x^2$ as $|x| \rightarrow \infty$ (see Figure 10). Compared to the family of 2D Sharp Kelvinlets, our Mixwell brush has an $O(1/r^2)$ fall-off and cusp singularity similar to the 2D Cusped Laplacian Kelvinlet (equation 25 in [de Goes and James 2019]); we can show that for $\varepsilon_{LapK} = \sqrt{3}\varepsilon$ the profiles asymptotically match cusps as $|x| \rightarrow 0$ and decay rates as $x \rightarrow \infty$. Beyond that, the functions are different. Furthermore, the Mixwell

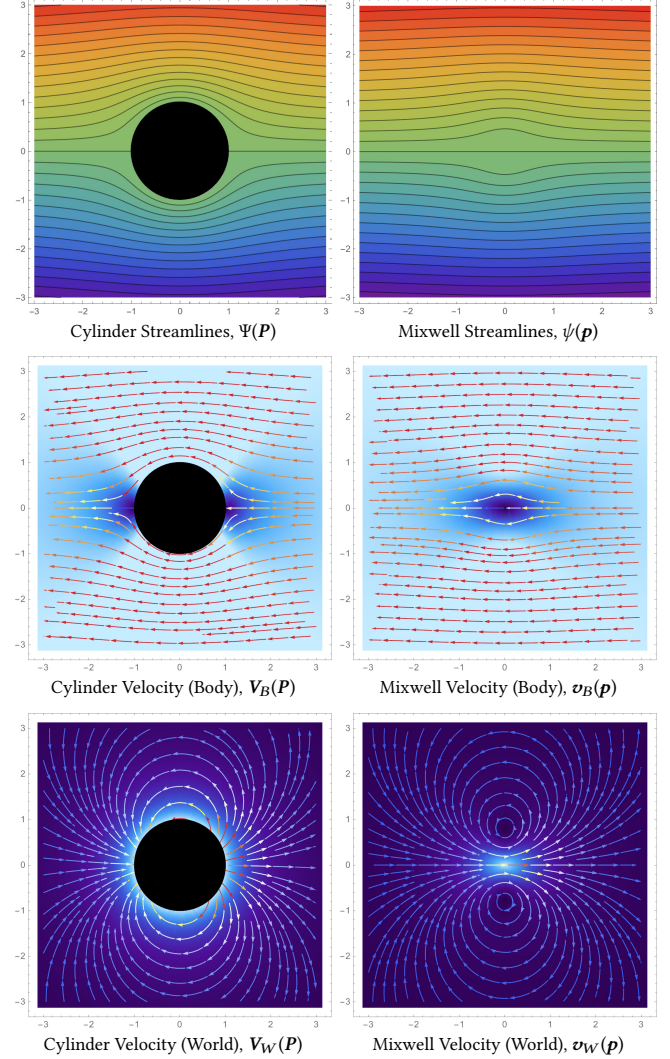


Fig. 9. **Flow Field Comparisons** for (Left column) Cylinder flow and (Right column) Mixwell flow. (Top) Flow streamlines of the cylinder and Mixwell stream functions colored by η_∞ values (unit cylinder radius, $\varepsilon = 1$). (Middle) Body-frame velocity fields of the cylinder and Mixwell flows showing flow stagnation points (in blue) responsible for streaking material through the fluid. (Bottom) Instantaneous world-frame velocity fields of the cylinder and Mixwell flow functions, showing unit speed (in white) at the tip of the cusped Mixwell brush. (A unit cylinder radius is shown here.)

brush is mathematically simpler without arctan evaluations, is the first cusped brush with a direct physical mixing derivation and interpretation, and it has fast GPU advection methods (finite and infinite strokes) and RDF machinery.

4.4 Affine Mixwell Brushes

Sharp pinch and twist brushes are derived using the affine brush approach of [de Goes and James 2017] in Appendix D. The scaling brush is zero for incompressible fluids, but it can be replaced by the

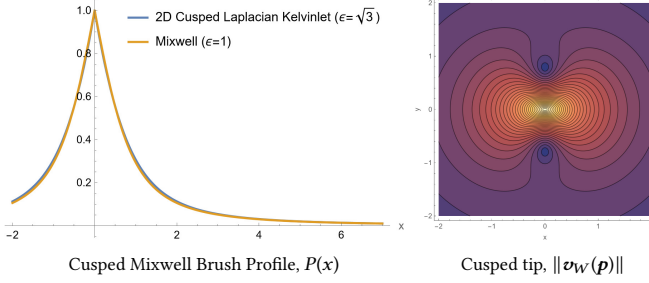


Fig. 10. **Sharp Mixwell Brush Profile:** (Left) The cusped tip of the Mixwell brush appears in its profile $P(x)$, versus a fitted 2D Cusped Laplacian Kelvinlet profile. (Right) A contour plot of the brush speed $\|v_W(\mathbf{p})\|$ shows the central cusp and zeros at $(x, y) = (0, \pm\sqrt{(\sqrt{5}-1)/2})$.

insertion/removal mapping (see Appendix C). Affine Mixwell brush deformations from our Shadertoy demo are shown in Figure 11.

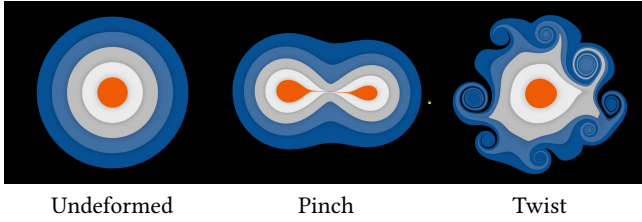


Fig. 11. **Affine Mixwell Brushes**

5 Reverse-Drift Function (RDF) Modeling

Inspired by the widespread use of Signed-Distance Fields (SDFs) [Jones et al. 2006; Quilez 2025] for geometric modeling in GPU pixel shaders [Quilez and Jeremias 2025], we propose RDFs as convenient displacement-field primitives for modeling texture advection.

5.1 Simulating Drift with the Mixwell Brush

Consider the quasistatic flow induced by a Mixwell brush of radius ε moving along the prescribed curve, $\mathbf{x}(t)$, $t \in [0, 1]$. The motion of a virtual particle at $\mathbf{p}(t)$ that starts at $\mathbf{p}(0) = \mathbf{p}_0$ is governed by the Mixwell brush initial value problem (IVP):

$$\frac{d\mathbf{p}}{dt} = \mathcal{K}_\varepsilon(\mathbf{p} - \mathbf{x}) \frac{d\mathbf{x}}{dt}, \quad t \in [0, 1], \quad (30)$$

$$\mathbf{p}(0) = \mathbf{p}_0.$$

By time-stepping this nonlinear ODE, we can approximate $\mathbf{p}(1)$, and thus determine the resulting drift, $\mathbf{p}(1) - \mathbf{p}(0)$. Forward brush motions result in forward drifts, and reverse brush motions yield reverse drifts.

5.2 Simulating Flows in Reverse with RDFs

A 2D RDF is simply a displacement field, $\text{RDF}(\mathbf{p}) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, which describes a reverse-time flow process. Consider the 2D flow induced by an arbitrary forward movement in time of a cylindrical tine at another point of fluid $\mathbf{p}$ over time Δt . The usual particle advection process will show us where the particle $\mathbf{p}$ moves to $\mathbf{p}_{\text{fwd}}$. However, for image marbling, we want to know the reverse: what location

$\mathbf{p}_{\text{rev}}$ moved to $\mathbf{p}$, so that we can, e.g., look up the texture color at $\mathbf{p}_{\text{rev}}$. This process is well known in image deformation [Sims 1992] and is familiar from semi-Lagrangian fluid advection [Stam 1999], in which reverse-time advection is used to determine that $\mathbf{p}$ came from $\mathbf{p}_{\text{rev}}$. The RDF encodes this reverse-drift displacement $\mathbf{u}$ at $\mathbf{p}$:

$$\mathbf{u} = \mathbf{p}_{\text{rev}} - \mathbf{p} = \text{RDF}(\mathbf{p}). \quad (31)$$

Consequently, fluid flow actions encoded as RDFs can affect coordinate transformations for texture lookup using this simple pattern:

$$\mathbf{p} += \text{RDF}(\mathbf{p}). \quad (32)$$

The chaining of multiple fluid actions can now also be easily achieved. It follows that, given a forward sequence of multiple fluid-flow actions indexed as $i = 1, 2, \dots, n$, the effect on a point $\mathbf{p}$ can be determined by simulating the entire sequence in reverse using reverse-drift advection. In this way, we simulate the backward-in-time flow of a particle at $\mathbf{p}$ to the original reverse-drift location, $\mathbf{p}_{\text{rev}}$. Given the RDF of each step, $\{\text{RDF}_i\}_{i=1}^n$, we compose them in reverse order using $\mathbf{p} += \text{RDF}_i(\mathbf{p})$, from $i = n$ down to 1. In this way, we can model complex marbling patterns with RDFs. Since texture sampling is only performed at the end of this process, it avoids the intermediate blurring of advected textures [Sims 1992].

5.3 RDF for Infinite Strokes (rdLine)

Using the solution for an infinite stroke in §3.3.1, the 1D drift value $\xi(\eta_\infty)\hat{\mathbf{x}}$ corresponds to the reverse drift of a unit cylinder moving to the right along the x axis (recall that $\xi < 0$). We can generalize to a radius- ε cylinder moving in a unit direction $\hat{\mathbf{d}} = (d_x, d_y)$ passing through the origin, where the distance of the point $\mathbf{p}$ from the line (our η_∞) is $|\mathbf{p} \cdot \hat{\mathbf{d}}_\perp|$ with $\hat{\mathbf{d}}_\perp = (-d_y, d_x)$. The resulting line RDF is:

$$\text{rdLine}(\mathbf{p}, \varepsilon, \hat{\mathbf{d}}) = \varepsilon \xi \left(\frac{|\mathbf{p} \cdot \hat{\mathbf{d}}_\perp|}{\varepsilon} \right) \hat{\mathbf{d}}. \quad (33)$$

The preferred RDF implementation in our Mixwell library uses the fast series approximation of $\xi(\eta_\infty)$ in Appendix B.

5.4 RDF for Finite Strokes (rdSegment)

5.4.1 The rdSegment RDF. Consider a radius- ε tine moving along a line segment from $\mathbf{a} \rightarrow \mathbf{b}$. To compute the RDF at a point $\mathbf{p}_0$, we simulate the brush moving in reverse from $\mathbf{b} \rightarrow \mathbf{a}$ and time-step a fictitious particle moving in the Mixwell flow from $\mathbf{p}_0$ to $\mathbf{p}$, thus obtaining the reverse drift, $\mathbf{p} - \mathbf{p}_0$.

For numerical reasons, we can integrate the virtual particle's location in relative (body-frame) coordinates, $\mathbf{r} \equiv \mathbf{p} - \mathbf{x}$, instead of absolute world coordinates $\mathbf{p}$, to avoid floating-point cancellation for particles near the brush tip with $\mathbf{p} \approx \mathbf{x}$. The body-frame IVP is

$$\frac{d\mathbf{r}}{dt} = \mathcal{K}_\varepsilon^B(\mathbf{r}) \frac{d\mathbf{x}}{dt}, \quad t \in [0, 1], \quad (34)$$

$$\mathbf{r}(0) = \mathbf{p}_0 - \mathbf{x}(0).$$

The final drift is $\mathbf{p}(1) - \mathbf{p}(0) = \mathbf{r}(1) + \mathbf{x}(1) - \mathbf{p}_0$. Here $\mathcal{K}^B = \mathcal{K} - \mathbf{I}$ is the analytical body-frame kernel with brush functions $(A_\varepsilon - 1, B_\varepsilon)$.

We use adaptive midpoint integration of the reverse-time Mixwell flow for line-segment brush motion (Algorithm 1). We employ an adaptive step size (Line 8) that allows for larger time steps farther from the brush, is spatially coherent across pixels, and accurately

resolves the cusped flow near the tip for high-resolution mixing. Error analysis versus the step-size scaling parameter, α , confirms acceptable behavior (see Figure 13); we use $\alpha = 1/10$ in our examples.

As due diligence, our supplemental material explores the semi-implicit solution given by (20)–(22), and describes in detail how to compute rdSegment using a custom Newton-based advection solver with elliptic functions.

Algorithm 1 rdSegment($p_0, \varepsilon, \mathbf{a}, \mathbf{b}$): Adaptive Midpoint Method

```

1: Input:  $p_0, \varepsilon, \mathbf{a}, \mathbf{b}$ 
2:  $\mathbf{x} = \mathbf{b}$  (Brush tip location)
3:  $\mathbf{r} = \mathbf{p}_0 - \mathbf{x}$  (Relative fluid-particle location)
4:  $L = \|\mathbf{a} - \mathbf{b}\|$  (Stroke length remaining)
5:  $\hat{\mathbf{L}} = (\mathbf{a} - \mathbf{b})/L$  (Stroke direction (reverse))
6: while  $L > 0$  do (Timestep reverse flow in body frame)
7:    $r = \|\mathbf{r}\|$ 
8:    $\Delta L = \min(L, \alpha \max(\varepsilon, r))$  (Adaptive stepsize)
9:    $\Delta \mathbf{x} = \Delta L \hat{\mathbf{L}}$  (Reverse brush motion)
10:   $\mathcal{K}_{mid}^B = \mathcal{K}^B\left(\mathbf{r} + \frac{\mathcal{K}^B(r)\Delta \mathbf{x}}{2}\right)$  (Midpoint method)
11:   $\Delta \mathbf{r} = \mathcal{K}_{mid}^B \Delta \mathbf{x}$  (Reverse particle drift)
12:   $\mathbf{r} += \Delta \mathbf{r}, \quad \mathbf{x} += \Delta \mathbf{x}, \quad L -= \Delta L$ 
13: end while
14:  $\mathbf{p}_1 = \mathbf{x} + \mathbf{r}$  (World-frame particle position)
15: Return  $(\mathbf{p}_1 - \mathbf{p}_0)$  (Reverse drift)
    
```

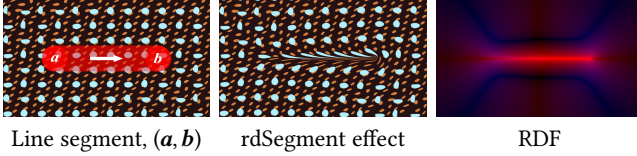


Fig. 12. **Plot of rdSegment RDF:** (Left) Line segment and brush radius ε producing $\mathbf{a} \rightarrow \mathbf{b}$ flow; (Middle) rdSegment flow; and (Right) visualization of RDF displacement field. Test rdSegment settings: $\varepsilon = 0.125$, $\mathbf{a} = (-0.5, 0)$, $\mathbf{b} = (0.5, 0)$, $\alpha = 0.1$ step-size parameter, and $\mathbf{x} \in [-1, +1]$ viewport.

5.4.2 Localization Masks. Complex patterns may have thousands of rdSegment strokes that would all require computation at every pixel sample. Fortunately, these displacement fields have a reasonably fast $O(1/r^2)$ fall-off with distance r ; for example, increasing the distance by a factor of 10 causes a 100-fold reduction. In our implementation, we optionally set a cutoff distance, $d_{\max} = \gamma \varepsilon$ beyond which the RDF is treated as zero; for complex comb patterns, we use $\gamma = 10$. We use the line-segment SDF with a smoothstep falloff to enforce C^0 continuity of the RDF.

5.5 Composing RDFs for Polyline Paths and Shapes

We can build RDFs for more complex shapes using rdSegment, such as polygons, triangle waves, spirals, or general polyline curves. In addition to chaining a sequence of points together in reverse order, we can also alternate the order of segments to create a broken line effect. See Figure 14 for triangle waves. By skipping some segments, we can also obtain a hatched curve. The appearance of closed-curve

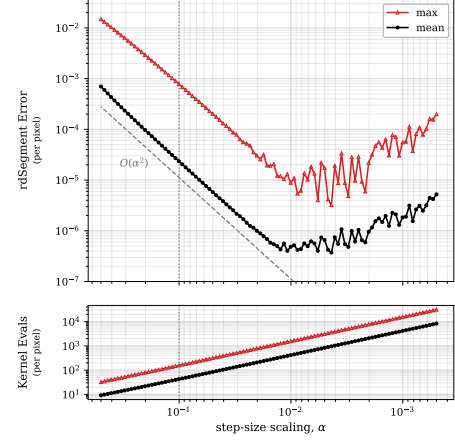


Fig. 13. **Accuracy of rdSegment RDF** in Algorithm 1 using adaptive midpoint method in body coordinates. Euclidean error is computed at each pixel an HD raster for the unit-stroke-length RDF test image (see Figure 12), using a 32-bit floating point Houdini-Copernicus OpenGL implementation. The mean RDF error over the image exhibits second-order convergence behavior with decreasing α step-size scaling parameter until the limits of the 32-bit floating point implementation are reached (near $\alpha \approx 0.01$); the max RDF error occurs near the brush tip but is also acceptable. In practice, good speed-accuracy trade-off is observed for the $\alpha = 1/10$ default setting. Reference RDF solution was computed offline using body-frame ODE integration of the rdSegment IVP using an explicit Runge-Kutta method of order 8 (scipy DOP853 in float64 at $\text{rtol}=1\text{e-}12$) for a reference far below the float32 noise floor.

RDFs, such as for a triangle, will depend on the starting and ending locations; it is possible to make them more symmetric by repeating the curve path or by using an approximate scheme based on the linear superposition of individual RDFs, which may not be physically accurate but may be sufficient for some applications.

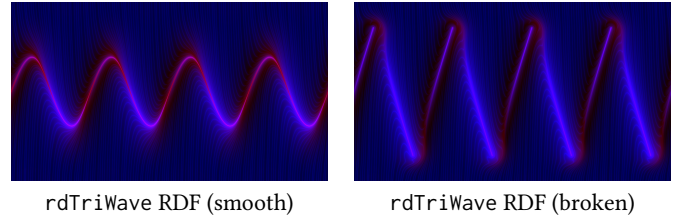


Fig. 14. A **Triangle Wave RDF (rdTriWave)** built from rdSegment zig-zags is a good approximation to waves used in popular patterns. The effect is used here to TriWave-marble a Nonpareil pattern (only RDF displacement field shown). (Left) For smooth waves the rdSegment RDFs are composed in reverse order, whereas for (Right) broken waves each $\mathbf{a} \rightarrow \mathbf{b}$ rdSegment is reversed to be $\mathbf{a} \leftarrow \mathbf{b}$ which produces a reverse-hatched wave.

5.6 RDFs for Oriented Curves

Curve RDFs are obtained by time stepping (34) with an adaptive step size on Δt , as in Algorithm 1 for rdSegment. In addition, a timestep restriction $\Delta t_{\max}$ is used to ensure that the curve shape is resolved. Our HLSL implementation of curve RDFs (rdCurve) uses templates,

with adaptive Euler (instead of Midpoint) to offset the cost of curve evaluation and resolution restrictions.

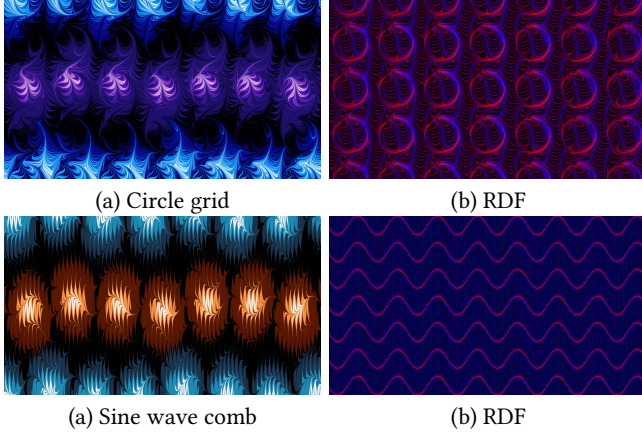


Fig. 15. **Various rdCurve RDF examples:** (Top) Circle grid; (Bottom) Sine-wave comb.

5.7 Comb and Array Modeling

We have modeled the flow around a single cylinder, which is a good approximation for an array of well-separated comb tines, but less so as they become closer. We now describe an approximation for computing drift using single-tine approximations for combs and arrays, and comment on the associated modeling error.

Line Combs. Combs consisting of multiple straight lines (rdLine) can have contributions summed in an order-independent manner due to the scalar linear-superposition nature of 1D drift. Given an evaluation point, we determine the maximum distance to contributing paths and sum their contributions, taking care to align any discontinuities at singular tine paths.

Combs consisting of multiple line segments. TriWave and related line-segment patterns compute nearby segments on a logical grid using mod and rounding tricks, together with segment SDFs (sdSegment) and a maximum influence distance from rdSegment. We then loop over these segments in reverse order to compose RDFs as needed. In practice, this is fast, with the major bottleneck being the average number of rdSegments that contribute to a pixel. Most patterns naturally limit the active segment range because tine radii are inherently small. For TriWave combs with multiple paths, we simulate each TriWave individually to ensure that the segments align end-to-end and avoid cross-wave interactions. For a given point, we then compose the contributing segment RDFs one wave at a time. Finally, we validated our RDF primitives (e.g., rdLine, rdSegment, and rdTriWave) against slow brute-force time-stepped cylinder flow models, and they matched the expected time-stepping error.

Combs and arrays of curves. More general arrays of finite curves, such as the circle grid comb (see Figure 15) are evaluated using the rdCurve primitive with a shape-specific SDF proxy. In our implementation, we use a bounding disk to provide inexpensive distance estimates for fall-off and masking. Combs of more general waves are possible, with sine waves shown in Figure 15.

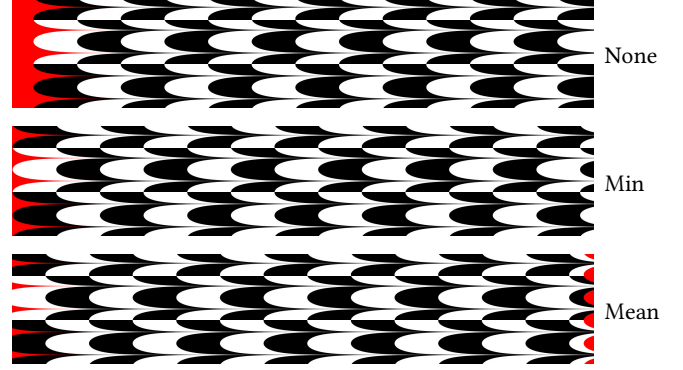


Fig. 16. **Comb Drift Compensation:** (Top) A Nonpareil comb applied to a rectangular checkerboard induces strong net drift, pulling substantial off-screen texture into view (red pixels). (Middle) Minimum-drift compensation has zero translation at the least-displaced locations, reducing red pixels to $\approx 33\%$, whereas (Bottom) mean-drift compensation further reduces them to $\approx 28\%$ but shifts the texture overall in the opposite direction.

5.8 Comb Drift Compensation

Combs with small tine pitch, H , can produce large average drifts, $O(1/H)$, that pull image-based art/textures offscreen (see Figure 16). We propose two drift compensations $\tilde{\xi}(\epsilon, H)$ to subtract from linear-superposition line combs:

- (1) **Mean-drift compensation:** Analogous to Darwin’s mean volume drift [Milne-Thomson 1996], the areal displacement of a single tine divided by the tine spacing, H , gives our mean-drift compensation:

$$\tilde{\xi}_{mean} = \frac{1}{H} \int_{-\infty}^{\infty} \left(\epsilon \xi\left(\frac{|y|}{\epsilon}\right) \right) dy = -\frac{\pi \epsilon^2}{H}, \quad (35)$$

which is significant as $H \rightarrow 0$.

- (2) **Min-drift compensation:** The minimum drift amount occurs at the midpoint between two adjacent tines, which we approximate by a truncated sum of tine influences. Using Mathematica, we derived the Padé approximation about $\hbar \equiv H/\epsilon$ equal to 2 for the nearest six tines:

$$\tilde{\xi}_{min} \approx \epsilon \frac{-37.7829 + 1.01689\hbar}{1.09635 + \hbar(12.2129 + \hbar(1.59101 + \hbar))}, \quad (36)$$

which is accurate to $\approx 0.002\epsilon$ for $H \geq 2\epsilon$.

In practice, either of these $\tilde{\xi}$ values can be subtracted from the comb drift to reduce overall translation (see Figure 16).

5.9 Transforming RDFs

Given an RDF primitive for a radius ϵ brush, $f(\mathbf{p}, \epsilon)$, common geometric transformations can be applied similarly to the case of SDFs. Translation by $\Delta\mathbf{p}$ is $f(\mathbf{p} - \Delta\mathbf{p}, \epsilon)$. Rotation by $\mathbf{R}$ about the origin is $f(\mathbf{R}^T \mathbf{p}, \epsilon)$. Scaling is slightly more complicated due to the brush radius ϵ . To geometrically scale the RDF shape by a factor $s > 0$, there are two cases: (1) $sf(\frac{\mathbf{p}}{s}, \epsilon)$ produces a physical zoom on a radius- ϵ RDF; and (2) $sf(\frac{\mathbf{p}}{s}, \frac{\epsilon}{s})$ scales the RDF’s shape but not the brush size. Finally, recall that in §5.3 we made the unit-radius 1D line RDF $\xi(y)$ exhibit radius- ϵ drift using the zoom, $\epsilon \xi(\frac{y}{\epsilon})$.

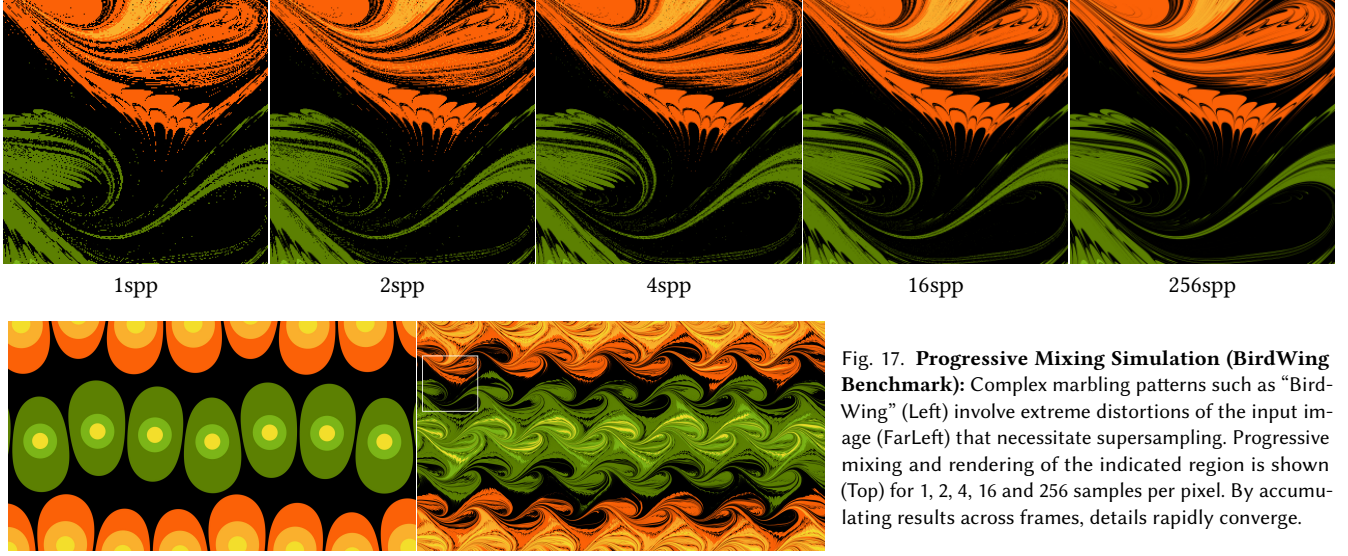


Fig. 17. **Progressive Mixing Simulation (BirdWing Benchmark)**: Complex marbling patterns such as “Bird-Wing” (Left) involve extreme distortions of the input image (FarLeft) that necessitate supersampling. Progressive mixing and rendering of the indicated region is shown (Top) for 1, 2, 4, 16 and 256 samples per pixel. By accumulating results across frames, details rapidly converge.

6 Progressive GPU Mixing and Rendering

Parallel shader-based drift evaluation enables fully progressive, per-pixel simulation and rendering of Mixwell phenomena, which we exploit in both offline and real-time settings.

6.1 Progressive Synthesis

Mixwell supports full-screen progressive fluid simulation and rendering as follows. We assume that positions and distances are expressed in units of pixel width. Given a pixel q with fragment coordinates $\mathbf{p}_q$, we jitter its location

$$\mathbf{p}_q \leftarrow \mathbf{p}_q + \text{random2D}(\text{frameN}) \quad (37)$$

using a shared, quasi-random sequence of samples on $[0, 1]^2$ for all pixels; in our implementation, we use Roberts’ R_2 sequence [Roberts 2018], which provides effective progressive sampling of both 1D and 2D patterns. We then compute its reverse drift $\mathbf{d}(\mathbf{p}_q) \in \mathbb{R}^2$ via

$$\begin{aligned} \mathbf{p} &= \mathbf{p}_q \\ \mathbf{p} &+= \text{RDF}_i(\mathbf{p}), \quad i = n \dots 1 \\ \mathbf{d} &= \mathbf{p} - \mathbf{p}_q. \end{aligned} \quad (38)$$

Our real-time HLSL-based DirectX application evaluates each RDF using a separate compute shader pass. This design primarily supports modularity, allowing arbitrary RDF sequences to be evaluated via corresponding sequences of compute shader dispatches. The resulting marbled coordinates are used to sample an input image (texture or shader) and progressively accumulate pixel colors.

Figure 17 shows the progressive synthesis of an expensive Bird-Wing pattern (§7.5.2). At 1920×1080 resolution, our application evaluates one sample per pixel in 10.4 ms on an NVIDIA GeForce RTX 4080 GPU, dominated (98%) by the two initial TriWave combs (10.2 ms); the remaining passes are inexpensive: noisy Nonpareil (0.10 ms), Gel-Git (0.05 ms), and accumulation (0.07 ms). Fortunately, we can do even better by exploiting the periodic RDF structure.

6.2 Fast Evaluation of Periodic RDFs

Many of the comb RDFs in this paper are periodic. For example, the 1D Nonpareil and Gel-Git uniform combs produce drift that repeats every pixel along the comb direction and every one and two time spacings, respectively, in the perpendicular direction. Implementations can take advantage of this property by evaluating image-based RDFs on the scale of their repetition rather than the entire image and tiling the results as necessary, provided the RDF’s period aligns properly with the pixels in the image.

6.2.1 Quantifying RDF Periodicity. Define a period vector $\mathbf{T} = (T_x, T_y)$ where $T_x, T_y \in \mathbb{N}^+ \cup \{\infty\}$. Different periods correspond to different canonical domains that we must compute (see Table 1). Here $(\mathbf{p} \bmod \mathbf{T}) = (x \bmod T_x, y \bmod T_y)$ denotes the componentwise Euclidean remainder, mapping $\mathbf{p}$ to the canonical domain $[0, T_x) \times [0, T_y)$; we use the convention that $z \bmod \infty = z$. In this notation, the periodicity condition becomes $\text{RDF}_k(\mathbf{p}) = \text{RDF}_k(\mathbf{p} \bmod \mathbf{T}_k)$. In practice, to compute a single periodic RDF, we only need to compute it on its canonical domain, which can provide a significant performance boost. The story is more complicated for the composition of multiple periodic RDFs.

Table 1. **Period vectors and canonical domains for RDFs.** W, H denote image dimensions.

Case	Period $\mathbf{T}$	Canonical Domain Size
Fully periodic	(m, n)	$m \times n$
1D function of x , periodic	$(m, 1)$	$m \times 1$
1D function of y , periodic	$(1, n)$	$1 \times n$
Periodic in x only	(m, ∞)	$m \times H$
Periodic in y only	(∞, n)	$W \times n$
No periodicity	(∞, ∞)	$W \times H$

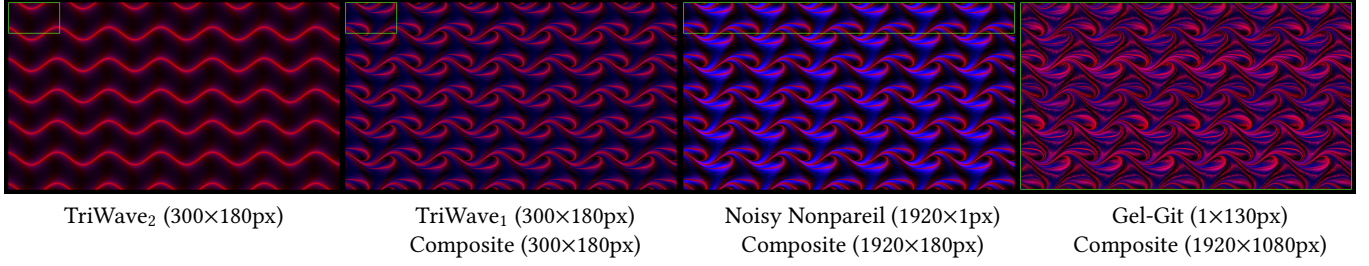


Fig. 18. **Period-Exploiting Evaluation of the BirdWing Pattern (Figure 17):** RDF passes are evaluated in reverse physical order (Left to Right) as $\text{TriWave}_2 \rightarrow \text{TriWave}_1 \rightarrow \text{Nonpareil} \rightarrow \text{Gel-Git}$. Each panel visualizes the composite drift map, and clarifies the period of (top label) the individual RDF pass T_k , and (bottom label) the resulting composite RDF image $\bar{T}_k$ (clamped to image size); rectangles indicate computed domains. Because earlier TriWave RDFs are expensive but periodic, they need only be computed on smaller domains (300×180 px), yielding significant cost savings over fullscreen (1920×1080) evaluation.

6.2.2 Period-exploiting Reverse RDF Composition. Although marbled point coordinates generally do not repeat, the *cumulative displacement* from the original pixel coordinate does. We therefore express reverse evaluation in terms of displacements and apply periodic wrapping at each pass. Initializing with the jittered point,

$$p_q \leftarrow p_q + \text{random2D}(\text{frameN}) \quad (39)$$

and $d = 0$, we have

$$\begin{aligned} d &+= \text{RDF}_n((p_q + d) \bmod \bar{T}_n) \\ &\vdots \\ d &+= \text{RDF}_1((p_q + d) \bmod \bar{T}_1) \\ p &= p_q + d. \end{aligned} \quad (40)$$

The key observation is that, after composing multiple periodic RDFs, the *accumulated* periodicity is not given by any single T_k , but by the componentwise **least common multiple** (lcm) of all periods composed so far¹: the period of the k^{th} composite drift map is

$$\bar{T}_k \equiv \begin{cases} \text{lcm}(T_k, \bar{T}_{k+1}), & k < n, \\ T_n, & k = n. \end{cases} \quad (41)$$

Thus, while intermediate passes may have smaller periods, it is the accumulated period that determines the canonical domain needed for period-exploiting evaluation (see Figure 18).

To see why, observe the recursive evaluation of the k^{th} pass:

$$d_k = d_{k+1} + \text{RDF}_k((p_q + d_{k+1}) \bmod T_k), \quad (42)$$

$$= D_k(p_q \bmod \bar{T}_k), \quad (43)$$

where D_k denotes the **composite drift map** at step k , stored as a 2D image over its canonical domain $[0, \bar{T}_k)$ —these are visualized in Figure 18. Since $d_{k+1} = D_{k+1}(p_q \bmod \bar{T}_{k+1})$, (42) shows that D_k depends on quantities with periods T_k and $\bar{T}_{k+1}$, and therefore, the smallest common period is T_k in (41). In implementation, the previous drift-map texture D_{k+1} is stored only on $[0, \bar{T}_{k+1})$, and all lookups use wrapped (modulo) addressing,

$$D_{k+1}(p) \equiv D_{k+1}(p \bmod \bar{T}_{k+1}). \quad (44)$$

The next composite drift map D_k is then evaluated for $p \in [0, \bar{T}_k)$

$$D_k(p) = D_{k+1}(p) + \text{RDF}_k((p + D_{k+1}(p)) \bmod T_k), \quad (45)$$

¹Recall that $\text{lcm}(a, b)$ is the smallest positive integer divisible by both a and b ; equivalently, $\text{lcm}(a, b) = ab/\text{gcd}(a, b)$ where $\text{gcd}(a, b)$ is the largest integer dividing both, e.g., $\text{lcm}(4, 6) = 12$. We adopt the convention $\text{lcm}(a, \infty) = \infty$ for $|a| > 0$.

where $D_{k+1}(p)$ uses the wrapped interpretation (44). This recurrence identifies $\bar{T}_k$ and provides the minimal canonical domain on which to compute D_k .

Consequently, $\bar{T}_k$ is nondecreasing as additional RDFs are composed (as k decreases), so later stages have larger periods and less redundancy (and potential for speedup). However, in practice, with reverse-ordered RDF evaluation, the most expensive combing passes, such as wave combs, occur early in this process and thus benefit most from periodicity exploitation. In contrast, cheaper 1D mixing passes like GelGit and Nonpareil typically occur last in the reverse-evaluation process and are not a bottleneck.

6.2.3 RDFs With Non-Integer Periods. For a variety of reasons (e.g., camera transform, RDF parameters) RDF periods may not properly align to a pixel grid, rendering tiling impossible. Evaluating multiple repetitions of the RDF can produce a period that aligns with the pixel grid again. The smallest pixel-aligned period is the least common multiple of the RDF's period in pixels and 1, $N = \text{lcm}_{\mathbb{Q}}(T, 1)$, where we define the *rational LCM* $\text{lcm}_{\mathbb{Q}}(T, 1)$ as the smallest positive integer divisible by $T \in \mathbb{Q}$. For a rational period $T = p/q$ in lowest terms, we have $N = p$ (achieved by tiling q repetitions). For example, if $T = 7/3$ pixels, we tile $k = 3$ repetitions to obtain the pixel-aligned period of $N = 3 \times \frac{7}{3} = 7$ pixels.

Since standard lcm requires integers, we use fixed-point arithmetic: with a scale factor s , let $T_s = \text{round}(T s)$ be the fixed-point representation of T . Then the pixel-aligned period is

$$N = \left\lfloor \frac{\text{lcm}(T_s, s)}{s} \right\rfloor. \quad (46)$$

In our implementation, we use $s = 10000$, which provides sufficient precision for typical RDF parameters. This extends naturally to 2D periods by applying (46) componentwise.

6.2.4 Periodic RDF Performance. By exploiting the periodic structure of comb RDFs, the BirdWing pattern from Figure 17 can be evaluated $21\times$ faster: 0.49 ms versus 10.4 ms previously. This is primarily due to exploiting TriWave periodicity. Timings are shown in the following table and in Figure 19.

BirdWing Passes	Canonical Domain Size (px)	Cost (ms)
Two TriWaves	300×180	0.35 ms
Noisy Nonpareil	1920×180	0.02 ms
Gel-Git	1920×1080	0.05 ms
Accumulate	1920×1080	0.07 ms

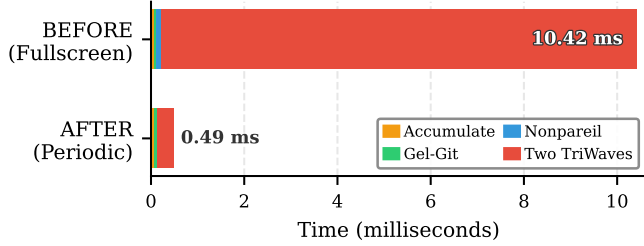


Fig. 19. **Periodic Rendering Performance** (BirdWing, 1920 × 1080, 1 spp): Exploiting periodicity produces $\approx 21\times$ speedup in each BirdWing pass.

7 Additional Results

7.1 Implementation Details

Our numerical methods are implemented in the Mixwell library, for which we have versions in both GLSL and HLSL, as well as OpenCL and OSL. Web brush demos will be provided in Shadertoy (GLSL), and most of the paper and video demos are generated using our real-time Windows HLSL application or the industrial-strength Houdini Copernicus (OpenCL) system (see Figure 20).

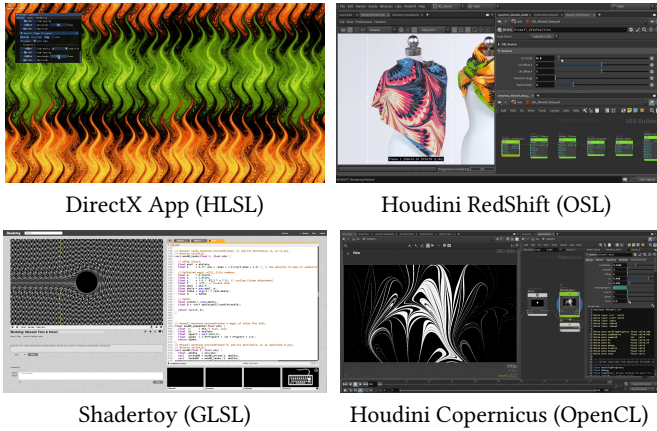


Fig. 20. **Four Interactive GPU Implementations of Mixwell**

7.2 Comparison to Newton-based Advection Solver

Our optimized Newton-based time-inversion advection solver, derived from Eqs. (20)–(22), is described in the supplemental document. With an optimized GPU implementation, it achieves interactive performance; however, it remains slower than our optimized Mixwell brush schemes, and it can encounter physical and numerical singularities that degrade visual quality in some cases.

On the Birdwing Benchmark (Figure 17), the two TriWave passes take the Newton-based solver an average of 68.2 ms, whereas the

Mixwell brush advection solver averages 10.2 ms. Relaxing the Newton tolerance can improve performance somewhat, but at the risk of visible artifacts. The dominant cost is the large number of special-function evaluations (XU) required per sample. This cost can remain high even in distant regions where the Mixwell brush effect is small and easily integrated.

7.3 Comparison to Stylus Brush of [Lu et al. 2012]

Geometric techniques have produced high-resolution brush-based image deformations from straight, time-line patterns [Lu et al. 2012, 2014]. Their stylus brush displaces a field point at distance d from the swept line or segment as $\frac{\alpha\lambda}{d+\lambda}\hat{u}$, with brush direction $\hat{u}$ and style parameters α, λ . As shown in Appendix E, in the infinitesimal-stroke limit this reduces to a Kelvinlet deformer with $B=0$, a classic “grab brush” [de Goes and James 2017] that is neither divergence-free nor area-preserving. The Mixwell brush, in contrast, is incompressible for both finite and infinite strokes (with consistent integrated responses) and produces the logarithmic drift singularity associated with cylindrical fluid strokes [Maxwell 1869]. See Figure 21 for a comparison. We use $\lambda = \varepsilon/2$.

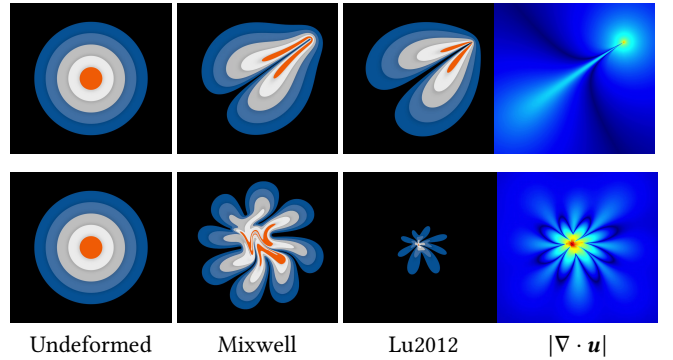


Fig. 21. **Comparison to stylus brush of [Lu et al. 2012]**: (Top Row) A single sub-stepped brush stroke with Mixwell and Lu2012 demonstrates the translational/grab-brush nature of the latter, with obvious divergence near the brush tip, and in its wake. (Bottom Row) Attempting to compress the bullseye with eight brush strokes is unsuccessful with Mixwell’s incompressible brush, whereas significant volume loss and divergence are evident in the Lu2012 case.

7.4 Approximating Rigid Boundaries

To approximate fluidic brushes in a rectangular 2D box, we approximate rigid slip boundary conditions (BCs) using a method of images technique [Milne-Thomson 1996]; more complex zero-velocity (no-slip) boundary conditions at a plane wall were considered for regularized Stokeslets in [Ainley et al. 2008]. For a single image brush, to ensure that the normal velocity component is zero on the edge, the source position $\mathbf{x}$ is mirrored to the image position $\mathbf{x}_i$, and we also reflect the brush displacement $\mathbf{b}_i = \mathbf{H}_i \mathbf{b}$:

$$\mathbf{u}(\mathbf{p}) = \mathcal{K}(\mathbf{p} - \mathbf{x})\mathbf{b} + \mathcal{K}(\mathbf{p} - \mathbf{x}_i)\mathbf{b}_i = \mathcal{K}(\mathbf{p} - \mathbf{x})\mathbf{b} + \mathcal{K}(\mathbf{p} - \mathbf{x}_i)\mathbf{H}_i\mathbf{b}.$$

For single edge reflections, $\mathbf{H} = \mathbf{I} - 2\mathbf{nn}^T$; for corner sources reflected once in both x and y directions, $\mathbf{H} = -\mathbf{I}$. In our examples, we use

eight nearby image sources (four edge and four corner) so that

$$\tilde{\mathcal{K}}(\mathbf{p}, \mathbf{x}) = \mathcal{K}(\mathbf{p} - \mathbf{x}) + \sum_{i=1}^8 \mathcal{K}(\mathbf{p} - \mathbf{x}_i) H_i. \quad (47)$$

Finally, we calibrate to match the brush-tip motion ($\mathbf{u}(\mathbf{x}) = \mathbf{b}$):

$$\mathbf{u}(\mathbf{p}) = \tilde{\mathcal{K}}_{\text{rigid}}(\mathbf{p}, \mathbf{x}) \mathbf{b} \equiv \tilde{\mathcal{K}}(\mathbf{p}, \mathbf{x}) (\tilde{\mathcal{K}}(\mathbf{x}, \mathbf{x}))^{-1} \mathbf{b}. \quad (48)$$

In practice, this approximates slip BCs well when the brush radius is small relative to the size of the domain (see Figure 22). Adding higher-order image brushes can improve BC accuracy but with diminishing returns.

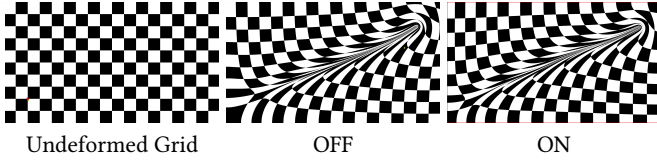


Fig. 22. **Approximate Rigid Slip BCs using Image Sources:** Using a truncated image-source construction, including single-reflection wall images and double-reflection corner images, we approximate rigid slip boundaries for small brushes. The OFF result shows freespace flow, while ON adds 4-edge and 4-corner image brushes and better approximates the rigid BC.

7.5 Simulation of Popular Marbling Patterns

We reproduce several well-known marbling patterns with our Mixwell implementation. While these patterns are not themselves technically novel, they provide a practical testbed for the tradeoffs of RDF composition and Mixwell advection in producing marbling-plausible flows. For reference, readers may compare against examples by professional artists (e.g., Galen Berry’s MarbleArt site [Berry [n. d.]]).

Animations of tine radius. Please see our video for related animations. We typically do not animate the comb’s physical motion through the fluid, since a stroke can be evaluated directly without time stepping. Instead, we animate the tine radius to illustrate how patterns vary with radius.

7.5.1 Base Patterns using Line Combs. Many foundational patterns are built from infinite straight-path comb strokes. These are especially fast in our system since they use only rdLine (with our series approximation). Two key examples are Nonpareil and Gel-Git.

Nonpareil. A uniform infinite comb stroke that yields an evenly spaced repetitive wave/ripple pattern and a common intermediate step (Figure 23). Small noise in tine spacing reduces regularity.

Gel-Git. (Chevron/Zig-Zag) Two staggered back-and-forth passes (Figure 24) produce strong mixing and are a common starting point for more complex patterns.

7.5.2 Complex Patterns using Finite Strokes. Many popular non-base patterns rely on wavy, finite comb strokes. We model these using a TriWave comb constructed from chained rdSegment RDF primitives. In practice, scenes often contain thousands of primitives, so we use localized RDF masking to restrict evaluation to affected regions and avoid per-pixel traversal of the full RDF list.

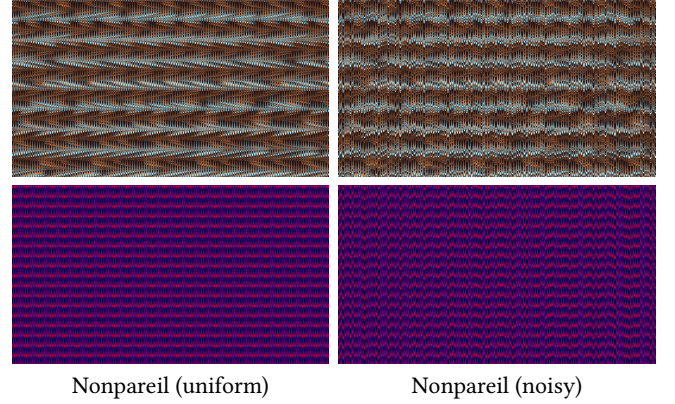


Fig. 23. **Nonpareil patterns** (Top) and RDF displacement field visualizations (Bottom) computed using a comb created from rdLine series approximation using (Left) uniform tine gaps, and (Right) noisy tine gaps which provide a more pleasing and natural look.

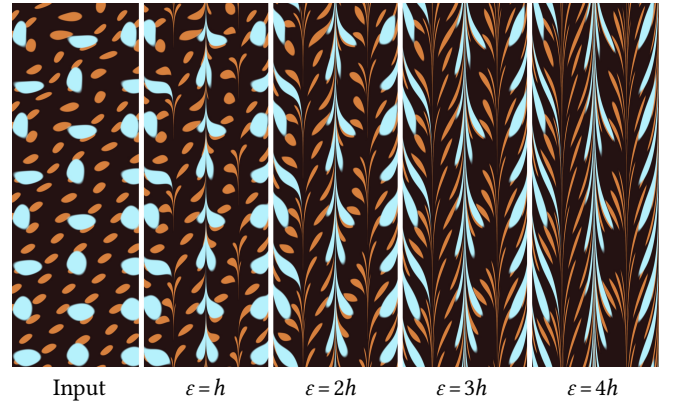


Fig. 24. **Gel-Git pattern** applied to an input texture using rdLine series combs, with (from Left to Right) an increasing tine radius, where $h = h_{\text{gap}}/10$ is one tenth of the comb’s gap. The Gel-Git pattern is often applied first to break up the input image structure.

Feather and BirdWing Patterns. Starting from a Gel-Git $\leftrightarrow$ base followed by a fine, noisy Nonpareil $\uparrow$ (Figure 23), a TriWave $\rightarrow$ pass produces the Feather pattern. A second TriWave $\leftarrow$ pass with reversed direction and inverted wave amplitude yields the BirdWing variant. Figure 25 visualizes both the marbled results and the fine displacement structure in the composed RDFs.

Peacock (Bouquet) and FrogFoot Patterns. These closely related patterns use the same finite-stroke TriWave construction but differ in the $\downarrow$ vs $\uparrow$ direction of the intermediate Nonpareil pass, which drives the characteristic “peacock” versus “frogfoot” appearance. Figure 26 shows the resulting images and their RDF structure.

Thistle and Tornado Patterns. We also reproduce two variants that swap the order of the orthogonal Gel-Git passes while keeping the same pair of downward TriWave passes. Despite the long finite strokes and many chained rdSegment primitives, the resulting displacement fields remain sharp and well resolved (Figure 27).

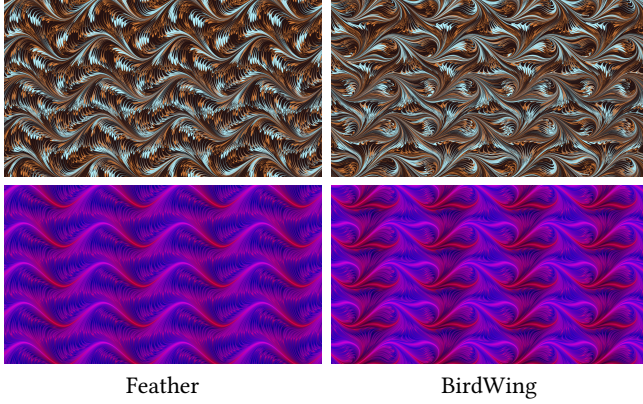


Fig. 25. **Feather and Birdwing Patterns:** (Left) Given a Nonpareil $\uparrow$ base pattern, we can approximate a Feather pattern using a TriWave $\rightarrow$ comb pass. (Right) The BirdWing pattern can be achieved using a second, inverted TriWave $\leftarrow$ pass. (Right) RDF visualizations show detailed displacement structures created by the Mixwell advection solver for the many chained rdSegment RDFs used in the TriWave comb RDFs.

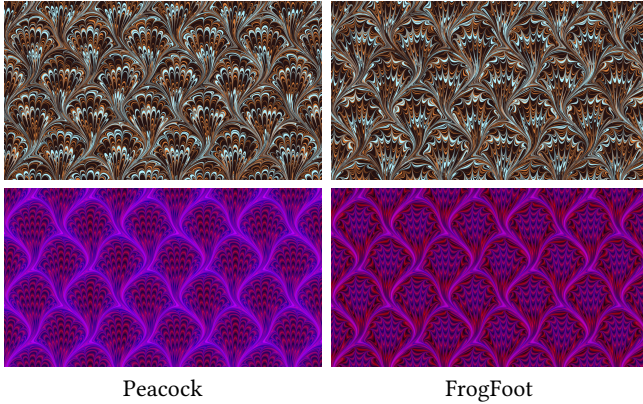


Fig. 26. **Peacock and FrogFoot Patterns:** Two closely related patterns that both start with Gel-Git $\leftrightarrow$ followed by either Nonpareil $\downarrow$ (Peacock) or Nonpareil $\uparrow$ (FrogFoot), and finally they both have two TriWave $\downarrow$ passes with the second pass in between and inverted. The direction of the Nonpareil is key to giving the characteristic Peacock or FrogFoot look. (Right) RDF visualizations show detailed displacement structures created by the Mixwell advection solver and the rdSegment-based TriWave comb RDFs.

Paint spread functions. Figure 28 visualizes paint spread functions for these patterns, highlighting the extreme, highly localized texture deformation produced by composed RDFs.

8 Conclusions and Discussion

We presented Mixwell, a family of sharp 2D fluid brushes and GPU-accelerated analytical drift methods for fast, resolution-independent mixing around a cylindrical tine. Our core technical contribution is a practical Maxwell drift evaluation pipeline for both infinite and finite strokes, enabled by a position-level advection formulation that avoids timestep accumulation, dissipation, and blur for long strokes. We further introduced cylindrical Reverse-Drift Functions

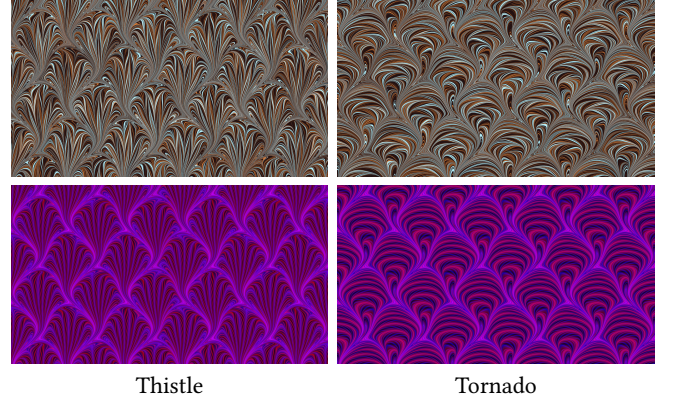


Fig. 27. **Thistle and Tornado Patterns:** (Left) Thistle starts with two Gel-Git passes as $\leftrightarrow$ then $\downarrow$, whereas (Right) Tornado uses two Gel-Git passes as $\downarrow$ then $\leftrightarrow$. The primary computational cost are the many rdSegments used to make up the two TriWave $\downarrow\downarrow$ comb passes.

(RDFs), a displacement-field abstraction that decouples stroke construction from texture evaluation and supports efficient composition of complex combing operations without intermediate resampling. We proposed progressive and period-exploiting real-time implementations in GLSL/HLSL and production integrations in Houdini (OpenCL/OSL), enabling arbitrary-resolution hydrographic mixing and marbling patterns with negligible numerical dissipation.

Limitations and Future Work

Mixwell is a specialized 2D solver for cylindrical tines, designed for sharp, controllable mixing rather than for general-purpose fluid simulation. Extending the underlying drift machinery to broader families of brushes (non-cylindrical cross-sections, anisotropic tips, or articulated multi-tine tools) is a natural next step but likely requires new analytical approximations and techniques.

Our model is based on incompressible potential flow, which enables efficient closed-form drift evaluation but omits inertial fluid dynamics. Techniques that rely on sloshing, waves, turbulence, or rapid tool motion are outside this regime. In addition, we treat the visual outcome through texture advection and do not model the material physics of real paints and inks (miscibility, surface tension, rheology, diffusion, pigment deposition, or film thickness). Bridging this gap will require experimental validation and richer material models, potentially combining Mixwell's drift fields with reduced-order or data-driven appearance models.

We also assume that tine effects can be composed of single-tine primitives. This becomes less accurate for dense combs, where near-field interactions between neighboring cylinders are significant, motivating multi-tine coupled brushes or interaction-aware approximations. One direction is to build compound Mixwell brushes using small linear constraint solves in the spirit of de Goes and James [2017]. More generally, uniform flow through periodic cylinder arrays can be approximated numerically (e.g., via BEM), while analytical and quasi-analytical approaches exist in restricted regimes [Crowdy 2016; Lamb 1924].

We found that advection based on the analytical Mixwell brush outperformed our Newton time-inversion approach (using the semi-implicit Maxwell drift-time integrals in the supplement), yielding faster, simpler, and more robust GPU evaluation without singularities. That said, improved analytical representations of the drift-time integrals may still make time inversion competitive in practice.

Acknowledgments

Doug James thanks Ken Museth and NVIDIA's High-Fidelity Physics Research group for early-stage feedback and support. We thank the anonymous reviewers for their feedback; Istinnaya (TurboSquid) for the teaser mannequin and scarves; Brittani Locke ("The Soft Asylum," <https://www.instagram.com/thesoftasylum>) and Galen Berry for marbling inspiration; Iñigo Quilez for Shadertoy resources; Houdini software courtesy of SideFX; Mathematica for derivations; Diligent Graphics for Diligent Engine; and ChatGPT and Claude for shader translation and writing feedback.

References

- Josephine Ainley, Sandra Durkin, Rafael Embid, Priya Boindala, and Ricardo Cortez. 2008. The method of images for regularized Stokeslets. *J. Comput. Phys.* 227, 9 (2008), 4600–4616.
- B. Tevfik Akgun. 2004. The digital art of marbled paper. *Leonardo* 37, 1 (2004), 49–52.
- Ryoichi Ando and Reiji Tsuruno. 2011. Vector graphics depicting marbling flow. *Computers & Graphics* 35, 1 (2011), 148–159. doi:10.1016/j.cag.2010.11.002
- A. Angelidis, M. P. Cani, G. Wyvill, and S. King. 2006. Swirling-sweepers: constant volume modeling. *Graph. Models* 68, 4 (2006), 324–32.
- Hassan Aref. 1984. Stirring by chaotic advection. *Journal of Fluid Mechanics* 143 (1984), 1–21. doi:10.1017/S0022112084001233
- Galen Berry. [n. d.]. The Art of Marbling – Marbled Paper & Marbling Supplies. <https://marbleart.us/index.htm>
- Kate Brett. 2021. *Making Traditional Marbled Papers*. The Crowood Press.
- B. C. Carlson. 1995. Numerical computation of real or complex elliptic integrals. *Numerical Algorithms* 10, 1 (March 1995), 13–26. doi:10.1007/BF02198293
- Anne Chambers. 1991. *Suminagashi : the Japanese art of marbling : a practical guide*. Thames and Hudson. <https://cir.nii.ac.jp/crid/1130282269416207744>
- Ricardo Cortez. 2001. The method of regularized Stokeslets. *SIAM Journal on Scientific Computing* 23, 4 (2001), 1204–1225.
- Darren G. Crowdy. 2016. Uniform flow past a periodic array of cylinders. *European Journal of Mechanics - B/Fluids* 56 (2016), 120–129. doi:10.1016/j.euromechflu.2015.10.003
- Charles G. Darwin. 1953. Note on hydrodynamics. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 49. Cambridge University Press, 342–354.
- Fernando de Goes and Doug L. James. 2017. Regularized Kelvinlets: Sculpting brushes based on fundamental solutions of elasticity. *ACM Trans. Graph.* 36, 4 (July 2017). doi:10.1145/3072959.3073595 Place: New York, NY, USA.
- Fernando de Goes and Doug L. James. 2019. Sharp Kelvinlets: Elastic deformations with cusps and localized falloffs. In *Proceedings of the 2019 Digital Production Symposium (DigiPro '19)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3329715.3338884
- Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. 2023. Fluid Simulation on Neural Flow Maps. *ACM Trans. Graph.* 42, 6 (Dec. 2023). doi:10.1145/3618392
- W. von Funck, H. Theisel, and H.-P. Seidel. 2006. Vector field based shape deformations. *ACM Trans. Graph. (SIGGRAPH Proc.)* 25, 3 (2006), 1118–1125.
- Mario Gazziero, João Paulo Gois, Candy Tenorio Gonzales, and José F. Rodrigues. 2018. A Computational Method for Interactive Design of Marbling Patterns. In *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 1–109. doi:10.1109/SBGAMES.2018.00010
- Guilherme G. Haetinguer and Eduardo S. L. Gastal. 2020. Regularized Kelvinlet Inversion for Real-Time Image Deformation and Video Time Warping. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 124–131. doi:10.1109/SIBGRAPI51738.2020.00025
- Xiaogang Jin, Shaochun Chen, and Xiaoyang Mao. 2007. Computer-Generated Marbling Textures: A GPU-Based Design System. *IEEE Computer Graphics and Applications* 27 (2007). <https://api.semanticscholar.org/CorpusID:863053>
- Mark W. Jones, J. Andreas Baerentzen, and Milos Sramek. 2006. 3D Distance Fields: A Survey of Techniques and Applications. *IEEE Trans. on Vis. and Comp. Graph.* 12, 4 (July 2006). Place: USA.
- B.-M. Kim, Y. Liu, I. Llamas, and J. Rossignac. 2007. Advections with Significantly Reduced Dissipation and Diffusion. *IEEE Trans. on Vis. and Comput. Graph.* 13, 1 (2007), 135–144.
- D. Kim, O. Song, and H.S. Ko. 2008a. A Semi-Lagrangian CIP Fluid Solver without Dimensional Splitting. In *Computer Graphics Forum*, Vol. 27. Blackwell Synergy, 467–475.
- Theodore Kim, Nils Thürey, Doug James, and Markus Gross. 2008b. Wavelet turbulence for fluid simulation. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 1–6. doi:10.1145/1360612.1360649 Place: New York, NY, USA.
- Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S Ebert, John P Lewis, Ken Perlin, and Matthias Zwicker. 2010. A survey of procedural noise functions. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 2579–2600.
- Horace Lamb. 1924. *Hydrodynamics*. University Press.
- John M Lee. 2013. *Introduction to Smooth Manifolds*. Springer.
- Shi-Guang Liu and Di Chen. 2012. Simulation Technique for Marbled Paper Patterns. *Journal of Software* 23, zk2 (2012), 1–7.
- F. Losasso, F. Gibou, and R. Fedkiw. 2004. Simulating Water and Smoke with an Octree Data Structure. *ACM Trans. Graph. (SIGGRAPH Proc.)* 23 (2004), 457–462.
- Shufang Lu, Aubrey Jaffer, Xiaogang Jin, Hanli Zhao, and Xiaoyang Mao. 2012. Mathematical Marbling. *IEEE Computer Graphics and Applications* 32, 6 (2012), 26–35. doi:10.1109/MCG.2011.51
- Shufang Lu, Xiaogang Jin, Hanli Zhao, and Yandan Zhao. 2013. Real-time image marbleization. *Multimedia Tools and Applications* 64 (June 2013). doi:10.1007/s11042-012-0989-0
- Shufang Lu, P. Y. Mok, and Xiaogang Jin. 2014. From design methodology to evolutionary design: An interactive creation of marble-like textile patterns. *Engineering Applications of Artificial Intelligence* 32 (2014), 124–135. doi:10.1016/j.engappai.2014.02.015
- Xiaoyang Mao, Toshikazu Suzuki, and Atsumi Imamiya. 2003. AtelierM: a physically based interactive system for creating traditional marbling textures. In *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE '03)*. Association for Computing Machinery, New York, NY, USA, 79–ff. doi:10.1145/604471.604489
- James Clerk Maxwell. 1869. On the Displacement in a Case of Fluid Motion. *Proceedings of the London Mathematical Society* s1-3, 1 (1869), 82–87. doi:10.1112/plms/s1-3.1.82 _eprint: <https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s1-3.1.82>
- James Clerk Maxwell. 1890. *The Scientific Papers of James Clerk Maxwell*. Vol. 2. University Press.
- Olivier Mercier, Xi-Yuan Yin, and Jean-Christophe Nave. 2020. The Characteristic Mapping Method for the Linear Advection of Arbitrary Sets. *SIAM Journal on Scientific Computing* 42, 3 (2020), A1663–A1685. doi:10.1137/18M1234424 _eprint: <https://doi.org/10.1137/18M1234424>
- Louis Melville Milne-Thomson. 1996. *Theoretical hydrodynamics*. Courier Corporation.
- Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. 2022. Covector fluids. *ACM Trans. Graph.* 41, 4 (July 2022). doi:10.1145/3528223.3530120
- Julio M Ottino. 1989. *The kinematics of mixing: stretching, chaos, and transport*. Vol. 3. Cambridge University Press.
- K. Perlin. 1985. An Image Synthesizer. In *Proc. of SIGGRAPH 1985*, 287–296.
- Ziying Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Trans. Graph.* 38, 4 (July 2019). doi:10.1145/3306346.3322945
- Iñigo Quilez. 2025. 2D Distance Functions: A collection of signed distance functions for 2D shapes. <https://iquilezles.org/articles/distfunctions2d/>
- Iñigo Quilez and Pol Jeremias. 2025. ShaderToy: Online GLSL Shader Editor. <https://www.shadertoy.com/>
- Martin Roberts. 2018. The unreasonable effectiveness of quasirandom sequences. *Extreme Learning* 575 (2018).
- A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac. 2008. An unconditionally stable MacCormack method. *J. of Sci. Comp.* 35, 2 (2008), 350–371.
- Mayumi Shōji, Hisashi Okamoto, and Takuya Ooura. 2009. Particle trajectories around a running cylinder or a sphere. *Fluid Dynamics Research* 42, 2 (Dec. 2009), 025506. doi:10.1088/0169-5983/42/2/025506
- Karl Sims. 1992. Choreographed image flow. *The Journal of Visualization and Computer Animation* 3, 1 (1992), 31–43. doi:10.1002/vis.4340030106 _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/vis.4340030106>
- Jos Stam. 1999. Stable Fluids. In *Proc. ACM SIGGRAPH*, 121–128.
- Jerry Tessendorf and Michael Kowalski. 2011. Resolution independent volumes. *Production Volume Rendering (SIGGRAPH 2011)* 2, 5 (2011), 12.
- 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.
- Adrien Treuille, Andrew Lewis, and Zoran Popović. 2006. Model reduction for real-time fluids. *ACM Trans. Graph.* 25, 3 (2006), 826–834.
- Sinan Wang, Yitong Deng, Molin Deng, Hong-Xing Yu, Junwei Zhou, Duowen Chen, Taku Komura, Jiajun Wu, and Bo Zhu. 2024. An Eulerian Vortex Method on Flow

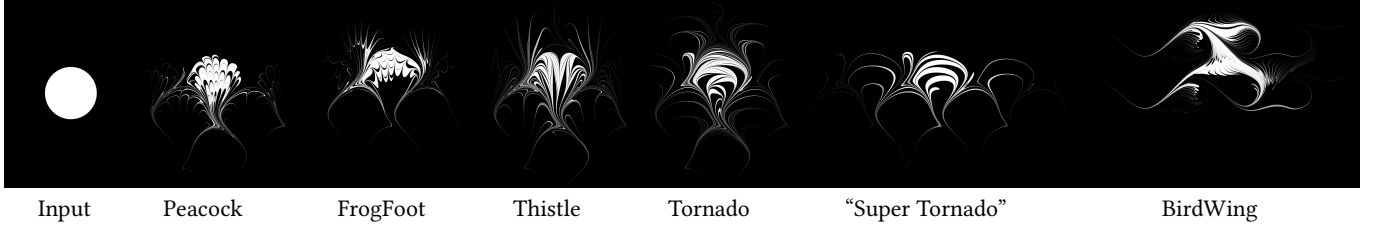


Fig. 28. **A Mixed-up Bunch of Paint Spread Functions:** Applying popular marbling patterns to a single white dot results in stark images that clearly show the extreme mixing and detail resolvable by Mixwell using super-sampled RDFs (4k images, 225 spp).

- Maps. *ACM Trans. Graph.* 43, 6 (Nov. 2024). doi:10.1145/3687996
- Daniel Weiskopf. 2004. Dye advection without the blur: A level-set approach for texture-based visualization of unsteady flow. In *Computer Graphics Forum*, Vol. 23. Wiley Online Library, 479–488.
- M Wicke, M Stanton, and A Treuille. 2009. Modular bases for fluid dynamics. *ACM Trans. Graph.* 28, 3 (2009), 1–8.
- D. C. Wiggert and E. B. Wylie. 1976. Numerical predictions of two-dimensional transient groundwater flow by the method of characteristics. *Water Resources Research* 12, 5 (1976), 971–977. doi:10.1029/WR012i005p00971
- Richard J Wolfe. 1990. *Marbled paper: its history, techniques, and patterns: with special reference to the relationship of marbling to bookbinding in Europe and the western world*. University of Pennsylvania Press.
- Takashi Yabe, Feng Xiao, and Takayuki Utsumi. 2001. The constrained interpolation profile method for multiphase analysis. *Journal of Computational physics* 169, 2 (2001), 556–593.
- Xi-Yuan Yin, Olivier Mercier, Badal Yadav, Kai Schneider, and Jean-Christophe Nave. 2021. A Characteristic Mapping method for the two-dimensional incompressible Euler equations. *J. Comput. Phys.* 424 (2021), 109781. doi:10.1016/j.jcp.2020.109781
- Jiayi Eris Zhang, Jérémie Dumas, Yun (Raymond) Fei, Alec Jacobson, Doug L. James, and Danny M. Kaufman. 2022. Progressive Simulation for Cloth Quasistatics. *ACM Trans. Graph.* 41, 6 (Nov. 2022). doi:10.1145/3550454.3555510 Place: New York, NY, USA.
- Jiayi Eris Zhang, Jérémie Dumas, Yun (Raymond) Fei, Alec Jacobson, Doug L. James, and Danny M. Kaufman. 2023. Progressive Shell Quasistatics for Unstructured Meshes. *ACM Trans. Graph.* 42, 6 (Dec. 2023). doi:10.1145/3618388 Place: New York, NY, USA.
- Yizhong Zhang, Chunji Yin, Changxi Zheng, and Kun Zhou. 2015. Computational hydrographic printing. *ACM Trans. Graph.* 34, 4 (July 2015). doi:10.1145/2766932 Place: New York, NY, USA.
- Hanli Zhao, Xiaogang Jin, Shufang Lu, Xiaoyang Mao, and Jianbing Shen. 2009. Ate-lierM++: a fast and accurate marbling system. *Multimedia Tools and Applications* 44 (2009), 187–203.
- Junwei Zhou, Duowen Chen, Molin Deng, Yitong Deng, Yuchen Sun, Sinan Wang, Shiyong Xiong, and Bo Zhu. 2024. Eulerian-Lagrangian Fluid Simulation on Particle Flow Maps. *ACM Trans. Graph.* 43, 4 (July 2024). doi:10.1145/3658180 Place: New York, NY, USA.

A Classical Analytical Derivation of Drift

The following is the derivation of particle drift following prior derivations [Darwin 1953; Maxwell 1869; Shōji et al. 2009]. By symmetry in space and time, it is sufficient to consider the part of the trajectory where $Y > 0$ and $-\infty < t \leq 0$, and at $t = 0$ we have $X_0 = X(0) = 0$ and $Y_0 = Y(0) = \eta_0 \geq 1$. Consider a value of $Y > 0$; then by $\Psi(X, Y) = \eta_\infty$ it follows that X is given (up to a sign) by

$$X^2 = Y(1 - Y(Y - \eta_\infty)) / (Y - \eta_\infty). \quad (49)$$

The time to flow from an $(X, Y) = (0, \eta_0)$ starting position to another can be determined by using

$$\dot{Y} = \pm 2(Y - \eta_\infty)^{3/2} \sqrt{1 - Y(Y - \eta_\infty)} / \sqrt{Y} \quad (50)$$

to integrate $dt = dY / \dot{Y}$ to obtain

$$t = -\frac{1}{2} \int_Y^{\eta_0} \frac{\sqrt{Y} dY}{(Y - \eta_\infty)^{3/2} \sqrt{1 - Y(Y - \eta_\infty)}} \quad (\eta_\infty < Y \leq \eta_0). \quad (51)$$

Then the absolute coordinate drift X_W of the particle from the starting location $X_W = 0$ is given by integrating

$$\dot{X}_W = \dot{X} + 1 = \frac{X^2 - Y^2}{(X^2 + Y^2)^2} = \frac{1}{Y} (Y - \eta_\infty) (1 - 2(Y^2 - \eta_\infty Y)) \quad (52)$$

to get

$$X_W = \int_0^t \dot{X}_W ds = -\frac{1}{2} \int_Y^{\eta_0} \frac{1 - 2(Y^2 - \eta_\infty Y)}{\sqrt{Y(Y - \eta_\infty)} \sqrt{1 - Y(Y - \eta_\infty)}} dY. \quad (53)$$

Although true, these expressions are somewhat messy with non-standard limits, and it is convenient to use the $+Y$ -axis angle $\sigma \in (-\pi/2, \pi/2)$ via the substitution:

$$\cos^2 \sigma = Y(Y - \eta_\infty), \quad (54)$$

which runs between $0 \rightarrow 1$ as Y goes $\eta_\infty \rightarrow \eta_0$. Here $|\sigma|$ goes $\pi/2 \rightarrow 0$ and is given by

$$|\sigma| = \arccos \sqrt{Y(Y - \eta_\infty)} = \arccos \sqrt{\frac{Y^2}{X^2 + Y^2}} = \arccos \frac{Y}{R} \quad (55)$$

using (8) and letting $R = \sqrt{X^2 + Y^2}$. The final expressions for $X_W(\sigma, \eta_\infty)$, $Y_W(\sigma, \eta_\infty)$ and $t(\sigma, \eta_\infty)$ are given in §3.2.2.

B Matched Asymptotic Expansion for 1D Drift

Our fast C^0 series approximation to the singular 1D drift function, $\xi(\eta_\infty)$ in (15), uses three Mathematica-derived high-order expansions in terms of the distance to the line, $y = |\eta_\infty|$:

(1) A singular degree-8 expansion about $y = 0$ captures the $\log(y)$ singularity:

$$\begin{aligned} x_{near}(y) = & -0.039720771 - 0.16369764 y^2 + 0.0076619254 y^4 \\ & - 0.00096477622 y^6 + 0.00015811568 y^8 \\ & + (0.50000000 + 0.093750000 y^2 - 0.0073242188 y^4 \\ & + 0.0010681152 y^6 - 0.00018775463 y^8) \log(y) \end{aligned} \quad (56)$$

(2) A smooth (5, 6)-Padé expansion about $y = 2.3$ (let $g = y - 2.3$) given by:

$$\begin{aligned} x_{mid}(y) = & (-0.0413839 - 0.0937878 g - 0.043355 g^2 - 0.0068946 g^3 \\ & + 1.77539 \cdot 10^{-6} g^4 - 1.01973 \cdot 10^{-7} g^5) \\ & / (1 + 3.26035 g + 3.66495 g^2 + 2.08615 g^3 + 0.665761 g^4 \\ & + 0.115772 g^5 + 0.00875686 g^6) \end{aligned}$$

(3) A far-field expansion with $O(1/y^3)$ decay:

$$x_{far}(y) = -\frac{\pi}{256} \left(\frac{64}{y^3} - \frac{192}{y^5} + \frac{600}{y^7} - \frac{1960}{y^9} + \frac{6615}{y^{11}} \right). \quad (57)$$

The final expression for the 1D drift (with a factor of 2) is ($y = |\eta_\infty|$):

$$\xi_{series}(\eta_\infty) = 2 \begin{cases} x_{near}(y), & y < y_{near} \\ x_{mid}(y), & y \in [y_{near}, y_{far}] \\ x_{far}(y), & y > y_{far} \end{cases} \quad (58)$$

where $y_{far} \equiv 6.50416858646776$ and $y_{near} \equiv 0.8220844420096408$ were numerically estimated to approximate a matched asymptotic expansion with C^0 continuity and low fitting error at matching points ($\approx 9 \cdot 10^{-6}$ at y_{near} , and $\approx 7 \cdot 10^{-8}$ at y_{far}). Keep in mind that this expansion is for the canonical unit-cylinder problem, so the reverse drift for the radius- ϵ case is $\epsilon \xi_{series}(\frac{\eta_\infty}{\epsilon})$ —thus, the matching error for a 1 mm radius tine is in the nm range. See Fig 29 for plots.

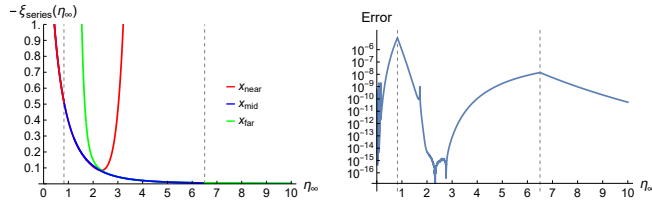


Fig. 29. **1D Drift Series Approximation**, $\xi_{series}(\eta_\infty)$: (Left) piecewise definition of $\xi(\eta_\infty)$ using three C^0 -matched expansions; (Right) plot of exceedingly low error between $\xi_{series}(\eta_\infty)$ and $\xi(\eta_\infty)$ in (15). All quantities are in units of the cylindrical tine radius, ϵ .

Regularizing the 1D Drift Singularity. The 1D drift function, $\xi(\eta_\infty)$ in (15), and its fast series approximation (§3.3.2), both exhibit a logarithmic singularity as $|\eta_\infty| \rightarrow 0$, which produces characteristic point streaking. In practice, we regularized the singularity by using $y \equiv \sqrt{\eta_\infty^2 + \delta^2}$; our implementation uses $\delta = 0.002$ to avoid NaNs.

C Modeling Tine Insertion and Removal

We model tine insertion and removal at the origin using an incompressible flow map with purely radial motion that conserves area in each annulus around the origin. Inserting the radius- ϵ tine introduces a disk of area $\pi\epsilon^2$, so a point at radius r expands to radius R with $\pi R^2 = \pi r^2 + \pi\epsilon^2$, and hence $R = \sqrt{r^2 + \epsilon^2}$ (cf. droplet insertion in [Lu et al. 2012]). Removing the tine reverses this process: a point at radius R contracts to radius r with $\pi R^2 = \pi r^2 + \pi\epsilon^2$, so $r = \sqrt{R^2 - \epsilon^2}$, and in particular, $R = \epsilon$ maps to $r = 0$ at the origin.

In summary, for a radius- ϵ tine at the origin, we use the following insertion and removal maps:

$$I_\epsilon : \mathbb{R}^2 \rightarrow \Omega_\epsilon, \quad I_\epsilon(\mathbf{p}) = \mathbf{p} \sqrt{1 + \frac{\epsilon^2}{\|\mathbf{p}\|^2}} \quad (\text{Insertion}) \quad (59)$$

$$I_\epsilon^{-1} : \Omega_\epsilon \rightarrow \mathbb{R}^2, \quad I_\epsilon^{-1}(\mathbf{P}) = \mathbf{P} \sqrt{1 - \frac{\epsilon^2}{\|\mathbf{P}\|^2}} \quad (\text{Removal}). \quad (60)$$

Numerical precautions must be taken when mapping points at the origin to the circle and back again. During insertion, any point with $r = 0$ is slightly perturbed to avoid division by zero. During removal, the square root should be taken robustly to avoid NaNs due to negative arguments.

The inverse Jacobian of the tine-insertion map is used for the velocity pullback (here $r = \|\mathbf{p}\|$ and $r_\epsilon = \sqrt{r^2 + \epsilon^2}$):

$$(DI_\epsilon(\mathbf{p}))^{-1} = \frac{r}{r_\epsilon} \mathbf{I} + \frac{\epsilon^2}{r_\epsilon r^3} \mathbf{p} \mathbf{p}^\top. \quad (61)$$

D Affine Mixwell Brushes

Twist and pinch brushes can be derived using the affine brush approach of [de Goes and James 2017] by making use of the brush's functions ($A_\epsilon(r)$ and $B_\epsilon(r)$) and their derivatives;

$$A'_\epsilon(r) = \frac{\epsilon^2(r^2 - 2\epsilon^2)}{r_\epsilon^5} \quad \text{and} \quad B'_\epsilon(r) = -\frac{\epsilon^2(4r^2 + \epsilon^2)}{r^2 r_\epsilon^5}, \quad (62)$$

and a 2×2 coefficient matrix, $\mathbf{F}$. The affine brush response is

$$\mathbf{u}_\epsilon(\mathbf{r}) = \sum_{ij} F_{ij}(\mathbf{r}) \partial_j \mathcal{K}_{:i}(\mathbf{r}) \quad (63)$$

$$= \frac{A'_\epsilon}{r} \mathbf{F} \mathbf{r} + \frac{B'_\epsilon}{r} (\mathbf{r}^\top \mathbf{F} \mathbf{r}) \mathbf{r} + B_\epsilon \mathbf{F}^\top \mathbf{r} + B_\epsilon \text{tr}(\mathbf{F}) \mathbf{r} \quad (64)$$

Pinch Brush ($\mathbf{F} = \mathbf{S}$). Here $\mathbf{S}$ is a traceless symmetric matrix representing a pinch and/or shear brush interaction:

$$\mathbf{u}_\epsilon^{(\text{pinch})}(\mathbf{r}) = \frac{B'_\epsilon}{r} (\mathbf{r}^\top \mathbf{S} \mathbf{r}) \mathbf{r} + \left(\frac{A'_\epsilon}{r} + B_\epsilon \right) \mathbf{S} \mathbf{r}. \quad (65)$$

Twist Brush ($\mathbf{F} = \tau \mathbf{R}_{90}$). The generalization of the cross-product matrix $[\hat{\mathbf{z}}]$ in 2D is $\mathbf{R}_{90} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$; it produces a 90° CCW rotation, $\mathbf{r}_\perp = \mathbf{R}_{90} \mathbf{r}$. The resulting twist brush for $\mathbf{F} = \tau \mathbf{R}_{90}$ is

$$\mathbf{u}_\epsilon^{(\text{twist})}(\mathbf{r}) = \tau \left(\frac{A'_\epsilon}{r} - B_\epsilon \right) \mathbf{r}_\perp. \quad (66)$$

Scaling Brush ($\mathbf{F} = s \mathbf{I}$). The scaling brush produces zero displacement, aligned with the fluid being incompressible. To add or remove fluid, we use the tine insertion/removal map from Appendix C.

E Lu2012 Stylus Brush is a Grab Brush

The stylus brush displacement [Lu et al. 2012, 2014] is

$$\mathbf{p}' - \mathbf{p} = \frac{\alpha \lambda}{d + \lambda} \hat{\mathbf{u}}, \quad (67)$$

where d is the minimum distance from $\mathbf{p}$ to the swept line or segment. The flow can match the brush-tip displacement $\hat{\mathbf{u}}$ under a no-slip condition with $\alpha = \|\hat{\mathbf{u}}\|$. In the infinitesimal-stroke limit, $d = r$, and (67) becomes a Kelvinlet deformer with $B = 0$:

$$\mathbf{u}(\mathbf{r}) = \left[\frac{1}{1 + r/\lambda} \mathbf{I} \right] \hat{\mathbf{u}} = A(\mathbf{r}) \hat{\mathbf{u}} = \mathcal{K}_\lambda^{Lu}(\mathbf{r}) \hat{\mathbf{u}}. \quad (68)$$

Because $B = 0$, this is a “grab brush” [de Goes and James 2017] that scales the input displacement with a distance-based fall-off; it is not divergence-free, so integration leads to a singular compressible deformation that is also inconsistent with the incompressible (and finite) infinite-line case (67).