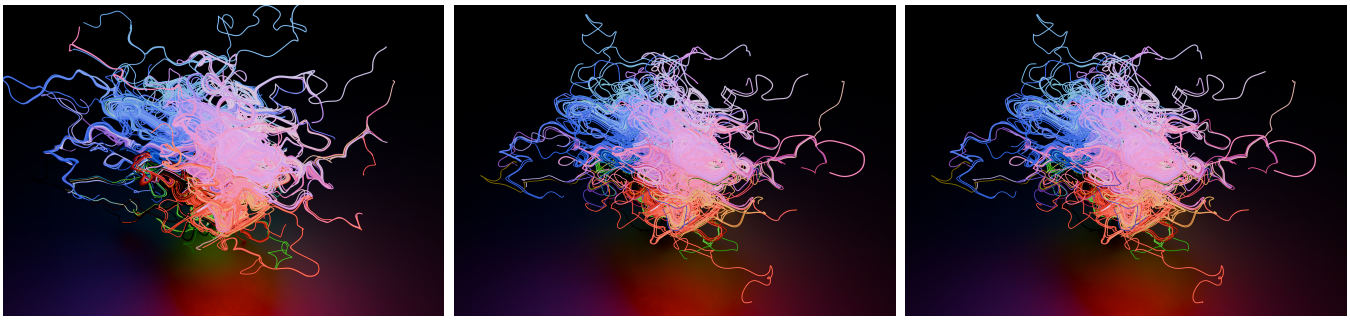
Improving Curl Noise

J. ANDREAS BÆRENTZEN, Technical University of Denmark, Denmark
JONÀS MARTÍNEZ, Université de Lorraine, CNRS, Inria, Loria, France
JEPPE REVALL FRISVAD, Technical University of Denmark, Denmark
SYLVAIN LEFEBVRE, Université de Lorraine, CNRS, Inria, Loria, France



a b c

Fig. 1. The luminous filaments are stream curves integrated along a divergence-free vector noise field. From left to right, the curves are computed using (a) plain Euler integration (300 steps), (b) Euler integration with our reprojection method (300 steps), and (c) fourth-order Runge-Kutta also with our reprojection method (600 steps, half step length). The last method is far more precise, yet (b) and (c) are nearly indistinguishable.

We introduce a divergence-free nD vector noise defined as the n-dimensional cross product of the gradients of $n-1$ noise functions. We show that this vector noise function is divergence-free and hence volume preserving for any dimension n. Our method enables precise integration and extends to new settings by substituting noise functions with implicit surfaces, (hyper)surfaces, or custom functions. We demonstrate applications including image warping, surface texturing, noise bounded by implicit surfaces, anisotropic curl-noise, and high-dimensional point jittering up to 7D.

CCS Concepts: • **Computing methodologies** → **Procedural animation**; *Texturing*.

Additional Key Words and Phrases: procedural noise, curl noise, divergence-free, vector fields

ACM Reference Format:
J. Andreas Bærentzen, Jonàs Martínez, Jeppe Revall Frisvad, and Sylvain Lefebvre. 2025. Improving Curl Noise. In *SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25), December 15–18, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3757377.3763980

Authors' Contact Information: J. Andreas Bærentzen, Technical University of Denmark, Kongens Lyngby, Denmark, janba@dtu.dk; Jonàs Martínez, Université de Lorraine, CNRS, Inria, Loria, Nancy, France, jonas.martinez-bayona@inria.fr; Jeppe Revall Frisvad, Technical University of Denmark, Kongens Lyngby, Denmark, jerf@dtu.dk; Sylvain Lefebvre, Université de Lorraine, CNRS, Inria, Loria, Nancy, France, sylvain.lefebvre@inria.fr.

SA Conference Papers '25, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-2137-3/25/12
https://doi.org/10.1145/3757377.3763980

1 Introduction

Liquids are often considered to be *incompressible* and, while this is only approximately true, it is an important difference between liquids and, say, gases. Without physical simulation, we can mimic a key aspect of liquid by generating incompressible flows, commonly achieved using *divergence-free vector fields*, i.e., fields, $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^n$, such that $\nabla \cdot \mathbf{c} = 0$.

Advecting a geometric shape along a divergence-free vector field preserves its volume, even if the shape becomes highly distorted. Similarly, if particles are advected along a divergence-free vector field, they will never get stuck or be repelled from specific regions, as divergence-free vector fields contain neither sinks nor sources.

These properties can be used in a myriad of ways; for instance, we can model leaves or other light objects that move on the surface of a (somewhat) turbulent pond by moving them around using a 2D divergence-free vector field. This type of modeling requires only a vector field that is both noisy and divergence-free, and this has led to the emergence of methods known as *curl noise* [Bridson et al. 2007], which leverage the fact that the curl of a smooth vector field is itself divergence-free.

Bridson's original curl noise formulation, $\mathbf{c} = \nabla \times \mathbf{v}$, is tied to 3D vector fields as the curl operator is only defined in 3D. However, alternative approaches exist for defining divergence-free vector noise (DFVN). Our method builds on an idea by Ivan DeWolf [2005], who proposed generating 3D DFVN as the cross product of the gradients of two noise functions. This approach is appealing for several reasons: as DeWolf noted, it extends naturally to surfaces, and as we show in this paper, the cross product's generalization to nD allows the same DFVN formulation to be used in nD.

We also show that the cross product formulation enables a *reprojection* scheme that significantly improves integration accuracy

along the DFVN field. The advection of a point along a vector field corresponds to moving on an invisible track. The point will always veer off-track due to numerical error. Our method makes this track visible by aligning motion with iso-contours of the underlying noise functions, allowing us to reproject the point onto these contours, stay on track, and enhance precision.

Contributions. In summary, our contributions are:

- a generalization of cross product-based divergence-free vector noise to *n*D,
- a proof that the resulting vector field is divergence-free in any dimension, and
- a novel reprojection method that significantly improves integration accuracy along the DFVN field.

2 Related Work

Procedural noise is widely used in computer graphics to generate detailed textures efficiently, relying on compact algorithms with constant time and space complexity [Perlin 1985; Lagae et al. 2010]. While noise typically refers to scalar fields, several methods extend it to incompressible, noisy vector fields for simulating plausible flows – our work falls within this line of research.

To the best of our knowledge, Ivan DeWolf [2005] first introduced a method to generate divergence-free noisy vector fields by taking the cross product of the gradients of two noisy scalar fields. He also proposed replacing one gradient with a surface normal to produce a noisy vector field constrained to a surface. Although this procedural method has been overlooked for some time, it was recently rediscovered [Wu 2021]. Von Funck et al. [2006] constructed a divergence-free vector field, considering the cross product of two gradient fields for shape deformation. Bridson et al. [2007] introduced the so-called curl noise given by the 3D curl operator, and extended the formulation to handle boundaries – an approach further improved by Chang et al. [2022] and Ding and Batty [2023]. We revisit and generalize the methods of both DeWolf [2005] and Bridson et al. [2007] to *n*D, and introduce an enhanced advection scheme using an *n*D reprojection operator.

Curl noise has been widely adopted in academic research, notably for generating procedural turbulence in fluid, smoke, and fire simulations [Kim et al. 2008; Narain et al. 2008; Schechter and Bridson 2008; Horvath and Geiger 2009; Pfaff et al. 2012; Bridson 2015], as well as for interactive editing [Pan et al. 2013], sketching [Eroglu et al. 2018], parameter optimization of animations [Brochu et al. 2010], initializing velocity fields in confined fluids [Yang et al. 2019], and robust surface deformation [Brochu and Bridson 2009].

Beyond fluids, curl noise has been used to model noise-induced movements of insects in a swarm [Wang et al. 2014a] and general flocking behavior [Wang et al. 2014b]. It has enabled users to add eddies or turbulence at prescribed locations in a desertscape [Paris et al. 2019], acted as a vortex force acting on the thorax of simulated butterflies [Chen et al. 2022], and been applied to jitter lattice points for procedural blue noise sampling [Bærentzen et al. 2023].

Curl noise is also widely adopted in the industry, with native support in major software platforms such as Houdini, Maya, Unity, and Unreal Engine, as well as through third-party extensions.

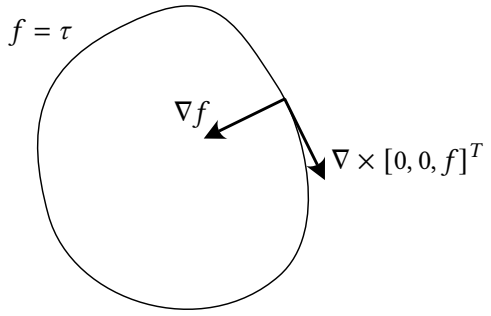


Fig. 2. For a scalar function f, the contour corresponding to iso-value τ is shown along with a gradient vector, ∇f, and $\nabla \times [0, 0, f]^T$

As demonstrated in prior work, curl noise is a powerful procedural technique for generating noisy, divergence-free vector fields with wide-ranging applications. Our approach unifies and extends existing methods through an improved advection integration scheme and unlocks new possibilities via an *n*-dimensional generalization of divergence-free vector fields, as illustrated in Section 5.

3 Background

In the following subsections, we define 2D and 3D curl noise and then present an extension to *n*D. As our starting point, we assume that we are in possession of a method for generating scalar noise functions [Perlin 1985], and that we can generate vector noise either by combining scalar noise functions or by taking the gradient of a scalar noise function.

3.1 Curl Noise

Given, a 3D vector field, $\mathbf{v} = [v^0\, v^1\, v^2]^T$, the curl, $\nabla \times \mathbf{v}$, is again a vector field,

$$\mathbf{c} = \nabla \times \mathbf{v} = \begin{bmatrix} \partial/\partial x \\ \partial/\partial y \\ \partial/\partial z \end{bmatrix} \times \begin{bmatrix} v^0 \\ v^1 \\ v^2 \end{bmatrix} = \begin{bmatrix} v^2_y - v^1_z \\ v^0_z - v^2_x \\ v^1_x - v^0_y \end{bmatrix} ,$$

where subscripts indicate partial derivatives. It is trivial to show that $\mathbf{c}$ is divergence-free since

$$\begin{aligned} \nabla \cdot \mathbf{c} &= \begin{bmatrix} \partial/\partial x \\ \partial/\partial y \\ \partial/\partial z \end{bmatrix} \cdot \begin{bmatrix} v^2_y - v^1_z \\ v^0_z - v^2_x \\ v^1_x - v^0_y \end{bmatrix} \\ &= v^2_{yx} - v^1_{zx} + v^0_{zy} - v^2_{xy} + v^1_{xz} - v^0_{yz} = 0 \ , \end{aligned}$$

where the last equality is due to the symmetry of second derivatives (i.e. $f_{xy} = f_{yx}$) known as Schwarz's Theorem or Clairaut's theorem. Thus, if $\mathbf{v}$ is a smooth and at least twice differentiable noise vector field then $\mathbf{c}$ is a noisy but divergence-free vector field.

3.2 2D Specialization

Perhaps surprisingly, specializing curl noise to 2D is helpful in showing how it can be generalized to *n*D. To generate 2D curl noise, we need to find a vector field, $\mathbf{v}$, whose curl, $\nabla \times \mathbf{v}$, is perpendicular to the z-axis. Given a twice differentiable noise function, $f : \mathbb{R}^2 \to \mathbb{R}$, this can be achieved by forming a vector field $\mathbf{v} = [0, 0, f]^T$, since we observe that $\nabla \times [0, 0, f]^T = [f_y, -f_x, 0]^T$ is precisely a vector

field perpendicular to the z-axis [Bridson et al. 2007]. It is also worth noting that it is the gradient with respect to x and y rotated clockwise by a right angle as shown in Figure 2.

Thus, for 2D curl noise, a single scalar noise function f suffices since $\mathbf{v}$ only has a component in the z direction. Moreover, the 2D curl noise construction indicates a way to generalize curl noise both to the nD case and to (hyper)surfaces. Observe that $\nabla \times [0, 0, f]^T = \nabla f \times [0, 0, 1]^T$. In other words, at least in 2D, curl noise can be computed as the cross product between a gradient vector field and another vector field (in this case, a constant field), and, as we shall see, this generalizes to n dimensions.

3.3 A Different 3D Formulation

In 3D, we can define a divergence-free noise using two gradient vector fields, say ∇f and ∇g [DeWolf 2005]. Our noise vector field is

$$\mathbf{c} = \nabla f \times \nabla g = \begin{bmatrix} f_y g_z - f_z g_y \\ f_z g_x - f_x g_z \\ f_x g_y - f_y g_x \end{bmatrix} .$$

Computing the divergence, we obtain

$$\begin{aligned} \nabla \cdot \mathbf{c} \quad &= \quad f_{yx} g_z - f_{zx} g_y + f_y g_{zx} - f_z g_{yx} \\ &\quad + f_{zy} g_x - f_{xy} g_z + f_z g_{xy} - f_x g_{zy} \\ &\quad + f_{xz} g_y - f_{yz} g_x + f_x g_{yz} - f_y g_{xz} \\ &= \quad 0 \ , \end{aligned}$$

again, the last equality follows from the symmetry of second-order derivatives, resulting in pairwise cancellation of all terms.

Note that we can replace either f or g with, for instance, a distance function. In this case, $\mathbf{c}$ will be tangent to the iso-contours of the object represented by the distance field. However, f and g must be at least twice differentiable; for distance fields, this condition fails on the medial axis where the gradient is discontinuous. As DeWolf [2005] suggests, smoothly transitioning one of the functions from a distance field close to the surface to a pure noise function deep inside an object can be used to create surface bounded divergence-free vector noise.

As we have seen, divergence-free vector noise can be formulated both in terms of the curl of a single vector field and as the cross product of two gradient fields. The question naturally arises whether the two formulations are fundamentally equivalent. Put differently, can any vector field, $\mathbf{v} = \nabla \times \mathbf{w}$, also be expressed as the cross product of two gradient fields? This turns out not to be the case. Given a function $f : \mathbb{R}^3 \to \mathbb{R}$ and a vector field, $\mathbf{v}$, we seek a condition for whether $\mathbf{v}$ lies in the tangent plane of iso-contours of f, i.e. $\mathbf{v} \cdot \nabla \mathbf{f} = 0$. Our proposition is that a necessary condition is

$$\mathbf{v} \cdot (\nabla \times \mathbf{v}) = 0 \ .$$

In order to prove this, we first observe that since $\mathbf{v}$ is perpendicular to the gradient, we can always express $\mathbf{v}$ in terms of a cross product,

$$\mathbf{v} = \nabla f \times \mathbf{a} \ ,$$

for some vector field $\mathbf{a}$. Plugging in this expression, we get

$$\mathbf{v} \cdot (\nabla \times \mathbf{v}) = \mathbf{v} \cdot (\nabla \times (\nabla f \times \mathbf{a})) = \mathbf{v} \cdot ((\nabla \cdot \mathbf{a}) \nabla f - \Delta f \mathbf{a}) = 0 \ ,$$

where the second equality follows from the well-known vector triple product rule and the last equality follows from the fact that $\mathbf{v}$ is perpendicular to both ∇f and $\mathbf{a}$.

With this proposition (which is in no way novel but follows from the Frobenius Theorem pertaining to differential topology) we can construct a counter example. The vector field $\mathbf{v} = [y, z, x]^T$ is the curl of $\mathbf{w} = \frac{1}{2}[z^2, x^2, y^2]^T$, but $\mathbf{v}$ cannot arise as the cross product of two gradient vector fields since $\mathbf{v} \cdot (\nabla \times \mathbf{v}) = -x - y - z \neq 0$.

3.4 Dimensions Greater Than Three

While the notion of curl is inherently tied to 3D, our reformulation relies only on the cross product, which generalizes to nD [Shaw 1987] as the product of $n - 1$ vectors defined by

$$\text{cross}(\mathbf{v}_1, \ldots, \mathbf{v}_{n-1}) \cdot \mathbf{v} = \det([\mathbf{v} | \mathbf{v}_1 | \ldots | \mathbf{v}_{n-1}]) \ , \tag{1}$$

where $\mathbf{v}_i, \mathbf{v} \in \mathbb{R}^n$ and $\det(\cdot)$ is the determinant. It is clear from this definition that the right-hand side (RHS) must be zero if $\mathbf{v} = \mathbf{v}_i$, and it follows that the nD cross product is orthogonal to its arguments. For a broader discussion of how the properties of the cross product generalize, the reader is referred to Shaw [1987]. In practice, we can compute $\text{cross}(\mathbf{v}_1, ..., \mathbf{v}_{n-1})$ as the determinant on the right-hand side of Eq. (1) replacing $\mathbf{v}$ with a column of nD basis vectors [Brahim Belhaouari et al. 2025].

The 4D Case. The above procedure clearly leads to the familiar rule in 3D, and in 4D we obtain

$$\text{cross}(\mathbf{a}, \mathbf{b}, \mathbf{d}) = \begin{vmatrix} \mathbf{i} & a^1 & b^1 & d^1 \\ \mathbf{j} & a^2 & b^2 & d^2 \\ \mathbf{k} & a^3 & b^3 & d^3 \\ \mathbf{l} & a^4 & b^4 & d^4 \end{vmatrix} \ ,$$

where $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$, and $\mathbf{l}$ are the canonical 4D basis vectors. Continuing with this example, we note that computing the divergence is tantamount to replacing the basis vectors with partial derivatives, i.e.

$$\nabla \cdot \text{cross}(\mathbf{a}, \mathbf{b}, \mathbf{d}) = \begin{vmatrix} \partial/\partial x & a^1 & b^1 & d^1 \\ \partial/\partial y & a^2 & b^2 & d^2 \\ \partial/\partial z & a^3 & b^3 & d^3 \\ \partial/\partial w & a^4 & b^4 & d^4 \end{vmatrix} .$$

Instead of $\mathbf{a}, \mathbf{b}$, and $\mathbf{d}$, we can plug in the gradients of three noise functions, f, g, and h. It is now clear that if we form the 4D vector noise,

$$\mathbf{c} = \text{cross}(\nabla f, \nabla g, \nabla h) \ ,$$

our check for whether $\mathbf{c}$ is divergence-free amounts to computing the value of the following determinant:

$$\begin{vmatrix} \partial/\partial x & \partial f/\partial x & \partial g/\partial x & \partial h/\partial x \\ \partial/\partial y & \partial f/\partial y & \partial g/\partial y & \partial h/\partial y \\ \partial/\partial z & \partial f/\partial z & \partial g/\partial z & \partial h/\partial z \\ \partial/\partial w & \partial f/\partial w & \partial g/\partial w & \partial h/\partial w \end{vmatrix} . \tag{2}$$

One could check manually that this expression always evaluates to zero. Unfortunately, for $n = 4$, we have 24 terms in the determinant, and after computing partial derivatives, we arrive at 72 terms, making it tedious to do the example by hand.

The nD case. In the following, we analyze the structure of a single term in Eq. (2) generalized to the nD case. Based on this, we will demonstrate that, thanks to Schwarz's Theorem, each term is always matched by a term that is of opposite sign but otherwise identical – exactly as in the 3D case.

One way to compute the determinant is to sum the products of all elements in the diagonal for all permutations of the rows. Thus, we can write the determinant as follows

$$\det([\nabla|\nabla f_1|...|\nabla f_{n-1}]) = \sum_{p\in P} \sigma(p) \frac{\partial}{\partial x^{p_0}} \prod_{i=1}^{n-1} \frac{\partial f_i}{\partial x^{p_i}} \ , \qquad (3)$$

where f_i is the i^{th} scalar field (i.e. in the 4D case, $f_1 = f$, $f_2 = g$, and $f_3 = h$), P is the set of all permutations of tuples $\langle 1, ..., n\rangle$, p_i denotes the i^{th} index in the permuted order starting from zero for practical reasons, and $\sigma(p)$ is the sign of the permutation. The sign is 1 if p is an even permutation and -1 if it is odd.

We note that the first factor in each term of Eq. (3) is the partial derivative operator, and the remaining factors are partial derivatives of the scalar field functions. By the product rule of differentiation, we can replace each term by a sum of $n-1$ products, each of which has the form

$$\sigma(p) \left[\frac{\partial f_1}{\partial x^{p_1}} \cdot ... \cdot \frac{\partial^2 f_i}{\partial x^{p_0} \partial x^{p_i}} \cdot ... \cdot \frac{\partial f_{n-1}}{\partial x^{p_{n-1}}} \right] \ .$$

In other words, the determinant is a sum of products and each product contains a double derivative. Now, the crucial thing to observe is that if we switch the two rows p_0 and p_i, we obtain a new permutation and the new permutation must have the opposite sign of the previous one, since the permutation must change from even to odd or the opposite. It follows that the two permutations lead to two products which have the same factors but opposite signs. Certainly, the order of the partial derivatives in the second-order term is swapped, but again, the order of differentiation does not matter. In other words, the determinant is a sum of products, all of which come in pairs with opposite signs, hence sum to zero. Consequently, our nD noise construct is indeed divergence-free.

Proof using exterior calculus. The formalism of exterior calculus makes the proof even easier. We need to show that the divergence of the vector noise is zero. In terms of the exterior derivative d the gradient of a scalar function f is $\nabla f = (\mathrm{d}f)^\sharp$, where $\sharp$ is an isomorphism that maps 1-forms to vectors.

Our nD vector noise is now defined as the wedge product of $n-1$ gradients of scalar noise functions,

$$\mathbf{c}(\mathbf{x}) = \xi(\mathbf{x})^\sharp \ , \qquad (4)$$

where

$$\xi(\mathbf{x}) = \star(\mathrm{d}f_1 \wedge \cdots \wedge \mathrm{d}f_{n-1}) \qquad (5)$$

and f_i are the $n-1$ scalar noise functions that we use as input, while $\star$ denotes the Hodge dual which maps a k-form to an $(n-k)$-form. To be clear, Eq. (5) is exactly the same as the nD cross product except that ξ is a 1-form, which is why the isomorphism in Eq. (4) is needed to map it back to a vector.

Divergence (applied to ξ) can be expressed as $\star\mathrm{d}\star$ and to show that $\mathbf{c}$ is divergence free, we need to show that

$$\star\mathrm{d}\star\,\xi = (-1)^{n-1} \star\mathrm{d}(\mathrm{d}f_1 \wedge \cdots \wedge \mathrm{d}f_{n-1}) = 0 \ , \qquad (6)$$

since $\star^2 = (-1)^{n-1}$ when the degree of the form is $n-1$ and the space is Euclidean [Dray 1999]. The proof is now all but trivial. We apply the Leibniz rule to the derivative and obtain,

$$\begin{aligned} \star\mathrm{d}\star\,\xi &= \star\mathrm{d}(\mathrm{d}f_1 \wedge \cdots \wedge \mathrm{d}f_{n-1}) \\ &= \star \textstyle\sum_{i=1}^{n-1} (-1)^i \left[\mathrm{d}f_1 \wedge \cdots \wedge \mathrm{d}^2 f_i \wedge \cdots \wedge \mathrm{d}f_n\right] \\ &= 0 \ , \end{aligned} \qquad (7)$$

where the last equality follows from the fact that each term in the sum contains a (wedge) factor $\mathrm{d}^2 f_i = 0$.

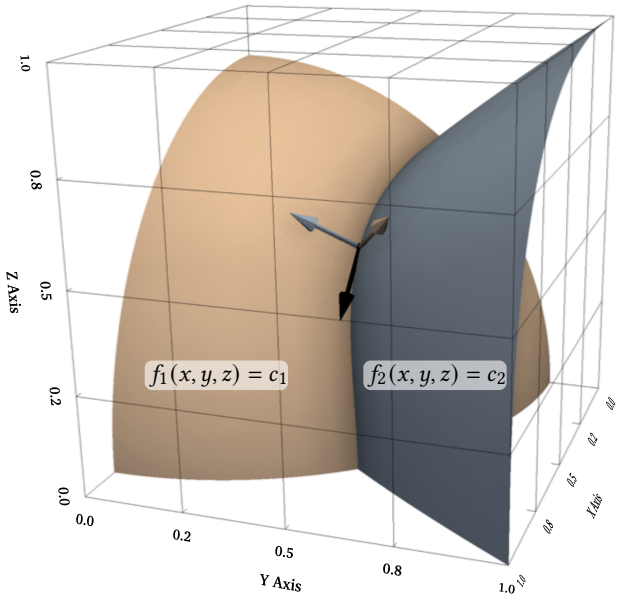


Fig. 3. Given $f_1(x,y,z) = x^2 + y^2 + z^2$, $f_2(x,y,z) = x^3 - y^2 + z^3$, and $\mathbf{p}_0 = (0.5, 0.5, 0.5)$, we plot the isosurface corresponding to $f_1(x,y,z) = c_1 = f_1(\mathbf{p}_0)$ in orange, and to $f_2(x,y,z) = c_2 = f_2(\mathbf{p}_0)$ in blue. At $\mathbf{p}_0$ we plot three arrows. The orange and blue arrows correspond to $\nabla f_1(\mathbf{p}_0)$ and $\nabla f_2(\mathbf{p}_0)$ respectively. The black arrow corresponds to $\nabla f_1(\mathbf{p}_0) \times \nabla f_2(\mathbf{p}_0)$.

4 Method

Our method requires the definition of $n-1$ scalar functions, $f : \mathbb{R}^n \to \mathbb{R}$, which are generally noise functions. In the region of interest, the f_i have to be at least C^2 continuous.

The divergence free vector noise is

$$\mathbf{c}(\mathbf{x}) = \mathrm{cross}(\nabla f_1(\mathbf{x}), ..., \nabla f_{n-1}(\mathbf{x})) \ . \qquad (8)$$

Given a starting point, $\mathbf{x}_0$, a point, $\mathbf{x}$, is propagated along the vector field by integration. The simplest possible method is Euler stepping,

$$\mathbf{x} \leftarrow \delta\mathbf{c}(\mathbf{x}) + \mathbf{x} \ , \qquad (9)$$

where δ is the step length, but in many of our experiments, we use the more precise and very popular fourth-order Runge-Kutta (RK4) scheme [Butcher 1996].

Improving precision. Of course, even more precise and sophisticated schemes for integration than RK4 could be used. However, since the specific vector field that we integrate is formed as the cross product of gradients, another option presents itself. It is well-known that for a smooth scalar field, f, the gradient, ∇f, at any point, $\mathbf{x}$, is perpendicular to the iso-contour, $\{\mathbf{y}|f(\mathbf{y}) = f(\mathbf{x})\}$, that passes

through $\mathbf{x}$. Since $\mathbf{c}$ is formed as the cross product of gradients, it follows that $\mathbf{c}$ lies in the tangent plane of iso-contours of all f_i (see Figure 3). It further follows that if we advect a point along $\mathbf{c}$ it will never leave the intersection of the iso-contours at its starting point if the integration is exact. In other words, the integral curves of $\mathbf{c}$ (its streamlines) are the intersections of the iso-contours of f_i. This suggests that we should constrain $\mathbf{x}$ to lie in the iso-contours given by the iso-values, $\boldsymbol{\tau} = \mathbf{f}(\mathbf{x_0}) = [f_1(\mathbf{x}_0), \ldots, f_{n-1}(\mathbf{x}_0)]$, of the initial point $\mathbf{x}_0$.

To this end, we introduce a reprojection step that iteratively moves $\mathbf{x}$ back onto all the isocontours. To solve this efficiently, we construct a first order Taylor approximation to $\mathbf{f}$ around $\mathbf{x}$ and form the linear system:

$$\mathbf{J}\mathbf{v} + \mathbf{f}(\mathbf{x}) = \boldsymbol{\tau} \ , \tag{10}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{f}(\mathbf{x})$ and we solve for $\mathbf{v}$. Consistent with the fact that we are projecting onto a linear approximation of a curve, the problem is underdetermined. This means that we have to find $\mathbf{v}$ as the minimum norm solution which we use to update $\mathbf{x}$

$$\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v} \ . \tag{11}$$

This process can be repeated until $\|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_0)\|$ is as small as desired or another stopping criterion is met.

As a simple alternative, reprojection can be performed by taking a Newton-Raphson step towards each iso-contour,

$$\mathbf{x} \leftarrow \mathbf{x} - \nabla f_i \frac{f_i(\mathbf{x}) - \tau_i}{\|\nabla f_i\|^2} \ . \tag{12}$$

In the 2D case where there is only a single function, f, the two approaches are equivalent. Refer to Section 5.2 and Figure 5 for an example of how this reprojection can dramatically increase the precision when DFVN is used as a tool for warping an image. In this example, we also compare Euler and RK4.

Algorithm 1 Divergence-Free Vector Noise Tracing in nD where f_i are the noise functions. The function takes starting point $\mathbf{x}_0$, integration time t, and number of steps, N, as arguments.

Require: $n, f_1, \ldots, f_{n-1}$
  function DFVN_TRACE($\mathbf{x}_0$, t, N)
    $\mathbf{x} \leftarrow \mathbf{x}_0$
    $\delta \leftarrow t/N$
    for $k = 1$ to N **do**
      $\mathbf{c} = \text{cross}(\nabla f_1, \ldots, \nabla f_{n-1})$
      $\mathbf{x} \leftarrow \delta\mathbf{c} + \mathbf{x}$
      $\mathbf{x} \leftarrow$ REPROJECT($\mathbf{x}, \mathbf{x_0}$)
    end for
    return $\mathbf{x}$
  end function

Algorithm 1 combines integration and reprojection in a procedure for tracing along a DFVN field. For simplicity, the algorithm uses an Euler step for integration, but this could be replaced with RK4 or something else. In Section 5.1 we explain how this algorithm can be applied to jittering of points in 3D through 7D.

DFVN on (hyper)surfaces. The functions f_i can be any twice-differentiable functions and are, thus, not required to be noise functions. This enables the construction of divergence-free vector noise on implicit (hyper)surfaces in nD using Algorithm 1. Specifically, one of the noise functions can be replaced with a function defining the (hyper)geometry. For instance, in 3D, curl noise can be generated on a surface by using a noise function as f_1 and a distance field (or any other implicit representation) defining the surface as f_2. This works because $\mathbf{c} = \nabla f_1 \times \nabla f_2$ lies in the tangent plane of the surface, hence the integral curves of the vector field lie in the surface. This application is discussed in Section 5.3 and an example is shown in Figure 6. It is important to note that f_2 must be twice differentiable on the surface. For a distance field this entails that the surface must have bounded curvature since the medial axis is otherwise arbitrarily close to the surface, and a distance field is not differentiable at points on the medial axis.

DFVN bounded by (hyper)surfaces. In an extension of the above scheme, we can generate divergence-free vector noise in a region bounded by a hypersurface. Again, we exemplify in 3D. Say, we have a noise function $\tilde{f_2}$. We now define

$$f_2(\mathbf{x}) = (1 - \alpha(\mathbf{x}))d(\mathbf{x}) + \alpha(\mathbf{x})\tilde{f_2}(\mathbf{x})) \ , \tag{13}$$

where $\alpha(\mathbf{x})$ is a function that goes smoothly from a value of 1 inside the surface and at a suitable distance from the boundary to 0 at the boundary and d is the signed distance to the boundary. Note that α in Eq. (13) must be a twice differentiable function of position and 1 at any point on the medial axis. Again, the surface must have bounded curvature to avoid that the medial axis is too close.

As $\mathbf{x}$ approaches the surface, f_2 approaches the distance field and far from the surface, it is simply a noise function. See Section 5.4 for applications of this.

Anisotropic DFVN through high-dimensional projection. An intriguing capability of our method is the ability to lift the problem to a higher-dimensional space, define a DFVN in that space, and then project it back to a lower dimension, thereby implicitly encoding properties such as anisotropy of the resulting vector field. For instance, to control anisotropy in 2D, one could lift to 3D by introducing a heightfield, construct the DFVN in this space, and then apply a simple 2D projection (see the result Section 5.5). While this is a straightforward example, more sophisticated variants, such as liftings to higher dimensions or more complex projection schemes to lower dimensions, are also possible. Overall, our method provides a way to introduce additional layers of control over the resulting vector field without requiring the construction of complex anisotropic procedural noise functions.

5 Experiments

Our DFVN formulation generalizes to higher dimensions, makes it straight forward to create vector noise on surfaces, and lends itself to precise integration, thanks to the reprojection scheme. In the following, we report on our experiments aimed at elucidating whether these properties lead to practical benefits. The experiments in Section 5.1 and Figure 1 employ the more precise reprojection scheme based on Eq. (10). In the remaining experiments, we use the simpler scheme based on Eq. (12).

Please find GLSL codes (suitable for ShaderToy) to replicate the results in Sections 5.2, 5.3, and 5.4 in the supplemental material associated with this paper. We also include a supplemental video containing animated comparisons. For the timings performed, we used a MacBook Pro with an Apple M1 Max SoC. The shader timings were computed as the median of 100 frames.

5.1 Point Sampling in nD

To test our DFVN formulation beyond 3D, we implemented curl noise jittering [Bærentzen et al. 2023] in a dimension-independent fashion. The core idea is to locally perturb points on a regular grid by advecting them along the noise vector field using a short time step. This disrupts the grid structure while avoiding clusters and gaps, as the divergence-free vector field contains neither sources nor sinks.

As the noise function, we use the simplex algorithm due to Perlin [2002] since its run-time complexity for a single lookup is polynomial rather than exponential in the number of dimensions (i.e., $O(n^2)$ rather than $O(2^n)$). Our implementation is a slightly modified version of the Python code by Craig Macomber[1] inspired by Gustavson [2005].

We implemented the following three point generation methods.

- Starting from a regular lattice of k^n points in the nD unit hypercube, points were jittered using Algorithm 1. We employed a single fourth-order Runge-Kutta step. Since the average distance to neighboring grid points increases with dimension, we used a time step proportional to the average distance, $\bar{d} = \sum_{i=1}^{n} 2^i \binom{n}{i} \sqrt{i}/(3^n - 1)$, from a point to its $3^n - 1$ neighbors.
- The same grid was used, but points were jittered randomly by sampling an offset vector from a uniform distribution over the interval $[-\frac{1}{2}u, \frac{1}{2}u]^n$ where $u = 0.1$ is the grid spacing.
- A collection of points was generated using Poisson Disk Sampling. This was done with SciPy, which employs the efficient implementation presented by Bridson [2007]. The original grid was not used for this experiment, but the grid spacing (u) was used as the disk radius.
- A collection of points was generated using the Sobol sequence generator in SciPy. For this experiment, the original grid was not used, and we rounded the number of points to the nearest power of two.

For all four methods, we generated point collections in 3D through 7D. For 3D, 4D, and 5D, we generated 10^n points. In 6D and 7D, we scaled back to 7^6 and 6^7, respectively.

Table 1 summarizes the main statistics from this experiment. For each dimension, we report the number of points used and the step length for integration, and we show the minimum and median distance between pairs of closest points, highlighting the greatest distance. To provide a more detailed comparison, we plot the radial distribution functions in Figure 4.

Curl noise jittering aims to generate point sets where no two points are too close, a characteristic of blue noise. Closest-point distances are thus a meaningful metric, and both radial distribution functions and median distances suggest that our method performs well even beyond three dimensions. Interestingly, in 7D, random jittering yields the highest minimum distance, despite allowing arbitrarily close pairs. This is expected, as in high dimensions, such close encounters become increasingly rare.

[1] https://github.com/Craig-Macomber/N-dimensional-simplex-noise

Table 1. This table summarizes statistics from the point collection generation experiment. Each entry contains the median distance followed by the minimum distance in parentheses and the wallclock time in seconds (after the colon). The largest values of median and minimum distance are highlighted in bold. Note that for the Sobol sequence, the actual number of points is rounded to the nearest power of 2.

n	CNJ	Jittering	PDS	Sobol
3	0.083 (0.044):1	0.069 (0.015):1	**0.104** (**0.100**):0	0.071 (0.031):0
4	0.081 (0.034):2	0.069 (0.010):1	**0.104** (**0.100**):3	0.074 (0.030):0
5	0.083 (0.023):21	0.073 (0.011):8	**0.103** (**0.092**):104	0.071 (0.019):10
6	**0.136** (0.039):31	0.118 (0.057):7	0.103 (**0.084**):651	0.078 (0.025):23
7	**0.176** (0.071):105	0.146 (**0.087**):20	0.102 (0.080):11k	0.081 (0.030):501

Table 2. This table compares the results of image warping using curl noise according to integration method and number of steps, whether reprojection was used (and the number of reprojection steps), the resulting frame rate (@1024x576), and RMSE between the images (@3840x2160).

		Euler		RK4	
steps	reprojection	FPS	RMSE	FPS	RMSE
64	no	158.262	8.671	70.388	6.625
64	1 step	105.269	7.685	53.717	3.861
64	10 steps	33.836	7.444	28.397	3.882
512	no	46.464	7.968	16.129	0.504
512	1 step	26.615	3.617	11.976	0.1
512	10 steps	4.241	2.777	3.558	0.098

In 3D–5D, Poisson Disk Sampling (PDS) outperforms all methods, with its median distance directly controlled by the disk radius (set to 1 in our experiments). However, more effective blue noise techniques exist in low dimensions, and PDS becomes increasingly inefficient in higher dimensions. As shown in Table 1, the runtime of PDS was over three hours for the 7D experiment whereas our method completed in less than two minutes. Note that the timings are ballpark numbers from a single run of these methods. Moreover, Jittering and CNJ are embarrasingly parallel – unlike PDS and Sobol – and were parallelized using joblib.

Finally, the influence of the gradients used in simplex noise [Gustavson 2005] is diluted in higher dimensions, leading to noise values closer to the function average (zero). This loss of magnitude is compounded by the nD cross product. To mitigate this, one could scale the noise function with a dimension-dependent factor. For our experiments (up to dimension 7) this compensation does not significantly change the results, but for higher dimensions it must be taken into account.

5.2 Image Warping

We can warp an image by applying Algorithm 1 to advect pixel coordinates along the vector field before looking up the value of the image. Results of this deformation on a simple wave image are

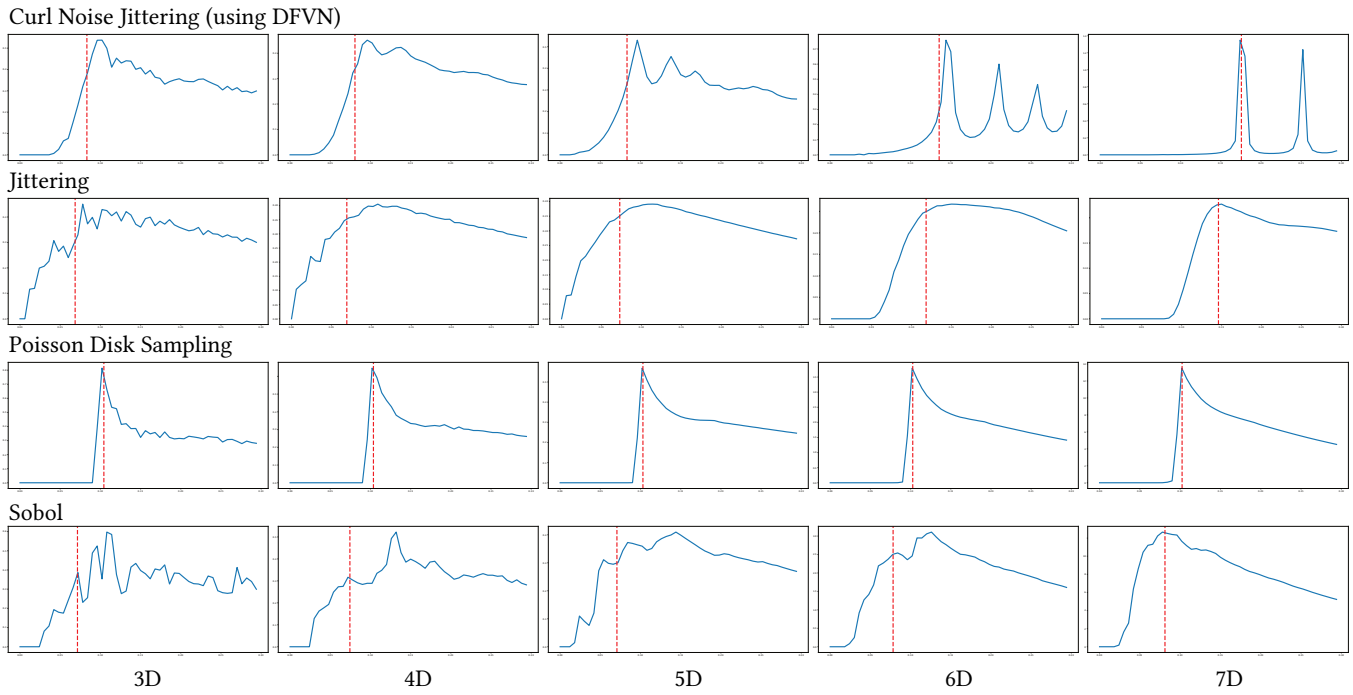


Fig. 4. Generation of point clouds in 3D through 7D. For each of the four methods (Curl Noise Jittering with DFVN, Jittering, Poisson Disk Sampling, and Sobol), we show the radial distribution function (RDF) (blue curve) and the median distance between a pair of closest points (stippled red line).

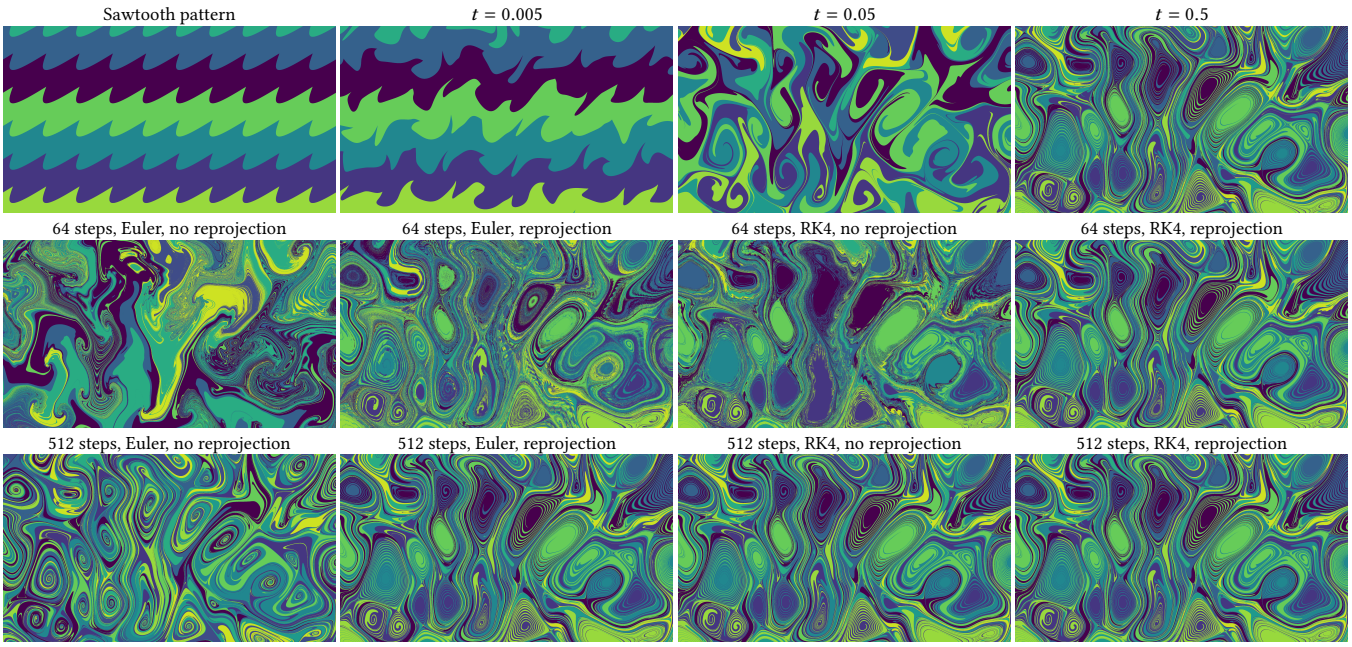


Fig. 5. Image warping using DFVN. The top row shows first the original wave pattern, and then the pattern after warping by flowing the image coordinates along the DFVN field for the indicated time t. The second and third rows show the image warped for time $t = 0.5$ using 64 and 512 integration steps, respectively. As indicated by the labels, the images were generated either with Euler integration or fourth-order Runge-Kutta and with or without Newton-Raphson based reprojection (12) onto the iso-contour of the noise.

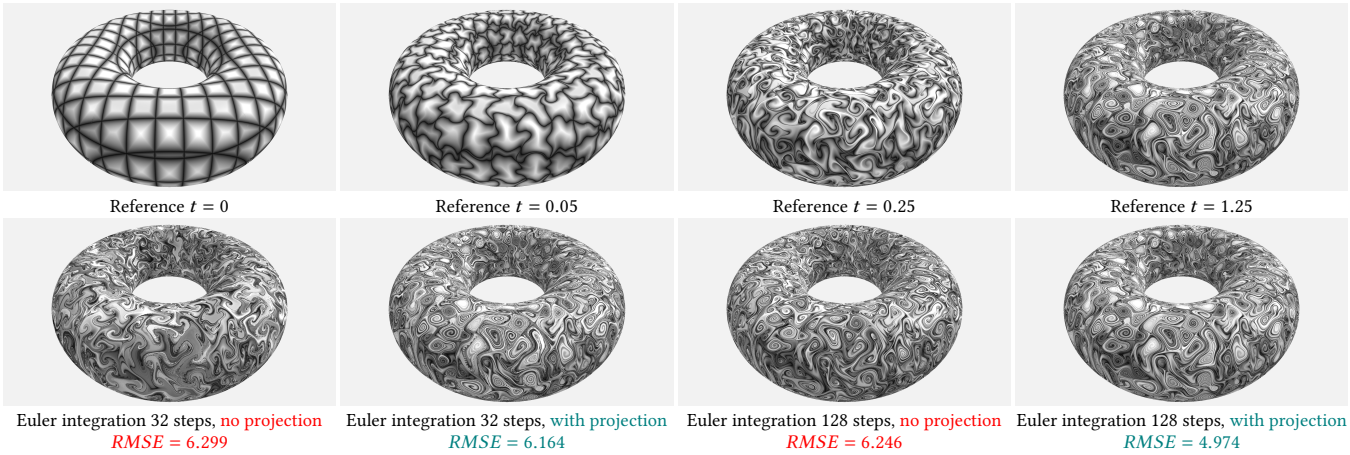


Fig. 6. On-surface texture warping with DFVN. *Top row*: reference results (2048 steps) for increasing time step t. *Bottom row*: results with fewer steps (32 or 128), with and without projection. Projection yields lower RMSE for the same number of steps, and the error decreases more rapidly as the step count increases.

shown in Figure 5. Our objective with this experiment is to show the importance of the reprojection scheme in the case of extreme warping using DFVN. Intuitively there should be well-defined swirls and no fold-overs. Since this is the 2D case, we reproject using Eq. (12).

Statistics for the experiment, including the root mean square error (RMSE) between each image and the reference image (top right in the figure), are summarized in Table 2. Note that in this table, we show the results for both a single reprojection step and ten steps of reprojection.

The first row of Figure 5 shows the effect of increasing integration time, using 1000 steps for the reference images. The second and third rows illustrate the impact of the integration scheme (Euler stepping versus fourth-order Runge-Kutta), varying the number of steps, and whether reprojection is used. We only show the result of a single step of reprojection since the images produced using one and ten steps are almost indistinguishable to the naked eye. Integration time $t = 0.5$ was used for all images in the second and third rows, and it is clear that many steps are needed to perform this warp.

The results are very unsurprising. All choices that should improve quality (RK4 instead of Euler, more steps, and using reprojection) have a clear positive impact on RMSE and a negative impact on frame rate. Notably, with our reprojection method, just 64 RK4 steps yield an image that is visually close to the reference.

5.3 Warping on a 3D Surface

Implicitly defined textures on surfaces can also be deformed, as shown in Figure 6. Similar to what was observed in Section 5.2, the reprojection variant consistently improves quality, yielding lower RMSE with minimal additional computational cost and lowering the RMSE at a higher rate as the number of integration steps increases.

5.4 DFVN Inside an Area or a Volume

Our method enables defining a DFVN confined within a smooth (hyper)surface. Figure 7 illustrates examples of manipulating the vectors in the cross product to contain the noise inside an area (Figure 7, top) or volume (Figure 7, bottom). In both cases, the first vector is a 3D noise gradient. To go from 2D to a contour (Figure 7, top), the second vector is interpolated from (0,0,1) inside to the normal of the contour. To go from 3D to a surface (Figure 7, bottom), the vector is interpolated from another noise gradient inside to the surface normal.

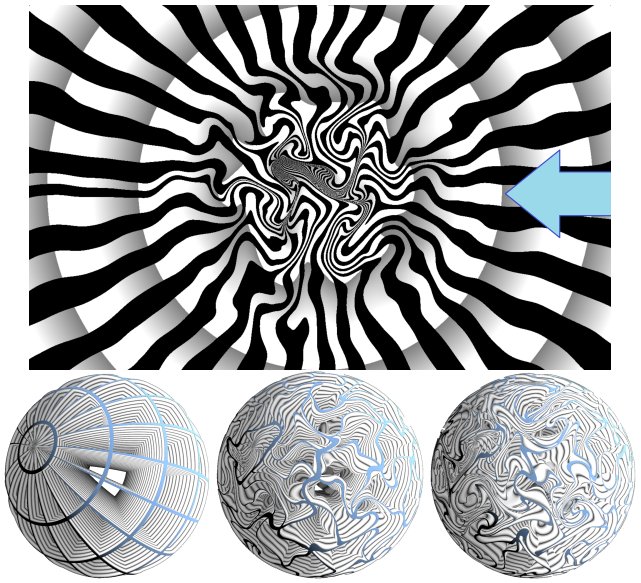


Fig. 7. *Top:* A radial pattern is warped by a noise progressively going from 2D (center) to constrained along a curve (circle indicated by the arrow). Beyond this circle, the noise is constrained along circular iso-contours, thus preserving the radii, only impacting the angles. *Bottom:* The same effect is applied to a ray-marched sphere. The noise on the sphere's surface is constrained, ensuring the blue outline is warped by a surface curl noise while the interior undergoes full 3D warping.

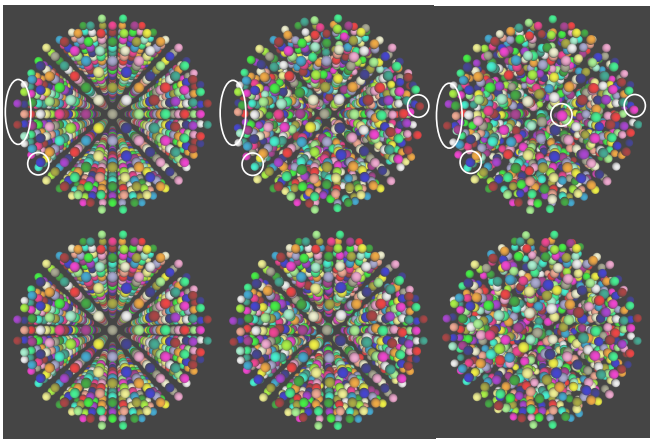


Fig. 8. Starting from particles on an octahedral lattice, we apply curl noise jittering [Bærentzen et al. 2023] in the top row and our boundary-conformant version in the bottom row. In both cases, the jittering does its job of achieving an irregular distribution, but note how particles in the encircled areas of the top row pop in and out of the domain (purple, brown, and blue particles on the left side of the image) or suddenly appear (pink particle in the center at frame 285). Since this popping is a temporal effect, it is seen much more clearly in the part of the supplemental video corresponding to this figure.

An interesting application of this ability of our method to do boundary-conformant DFVN is that we can do curl noise jittering without particles popping in and out of the domain of interest. As mentioned before, the purpose of curl noise jittering is to jitter particles away from initial vertices in a lattice to obtain an irregular distribution with blue-noise-like properties. If we use this concept to animate motion of particles, it is much preferred that the particles consistently stay within the domain of interest. This is illustrated in Figure 8 and the supplemental video. The curl noise jittering by Bærentzen et al. [2023] will keep the number of particles in the domain of interest close to a constant number, but the method has no tool for keeping the jittered particles within a boundary. With our method, this is simply achieved by replacing one of the noise functions (f_i) with the signed distance field of the (hyper)surface, or even better using Eq. 13.

5.5 Advection and Anisotropy Through Surface Design

Our method introduces anisotropy by lifting to a higher-dimensional space and then by projection back to the original domain. Figure 9 and Figure 10 illustrate how noises obtained from higher dimensions can be used to control effects in lower dimensions.

In Figure 9, particles are advected by a DFVN. Viewed from above – orthographic view in xy – the particles move from top to bottom in a motion perturbed by noise and a vortex in the center. This behavior results from defining the first vector in the cross-product as the gradient of a 3D scalar field, where the field has a vertical (z) ramp. The view looks down at a surface, which normally is used as the second vector in the cross-product. When the surface is a horizontal plane, the vertical ramp has no influence, and only the noise impacts the result. However, when the surface is sloped, the gradient in z interacts with the xy components of the surface normal. This introduces auxiliary motions, with the particles moving across the slope. This provides an intuitive and efficient way to control advection through surface design.

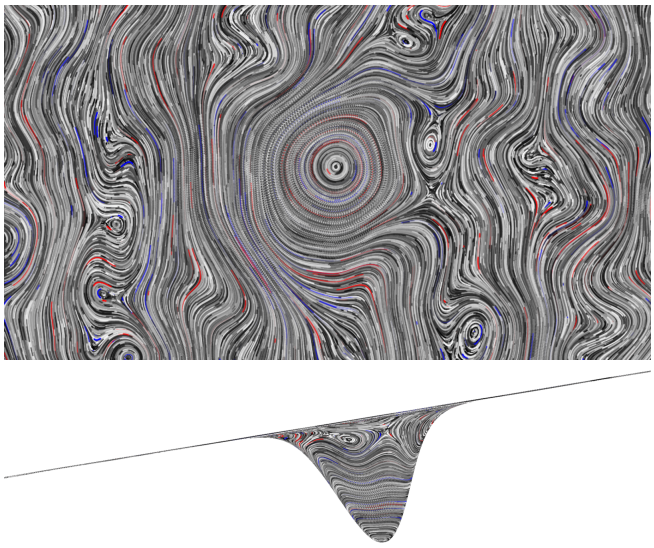


Fig. 9. *Top:* Traces of particles advected by a divergence free vector field. *Bottom:* Surface generating the advection; the slope normal induces motion in interaction with the gradient of the first field. The particle motion is orthogonal to the slope, and speed increases with slope.

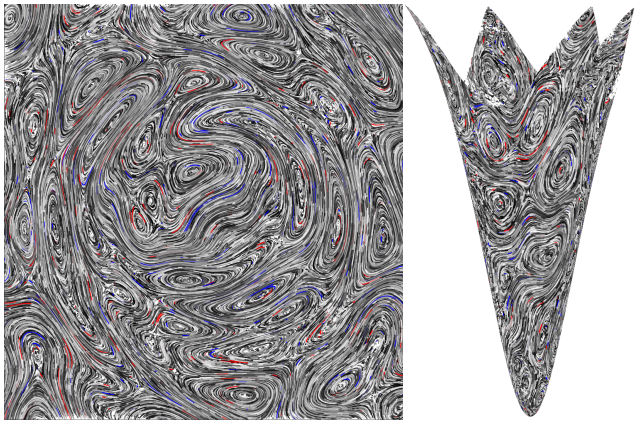


Fig. 10. *Left:* Traces of particles advected by a divergence free vector field. Note the overall anisotropic aspect of the vortices. *Right:* Surface generating the anisotropy.

Figure 10 demonstrates how the same technique can introduce anisotropy in the stream pattern. In this case, the field has no vertical ramp, and thus, the surface slope produces anisotropy without auxiliary motion. Both effects can be freely combined. Note that this is an application of DVFN rather than a way to form DFVN fields, and the result is only guaranteed divergence free before projection to the lower dimensional space (i.e. 2D in this case).

6 Discussion

We have shown that the formulation of divergence-free noise in terms of cross products of gradients of noise functions generalizes to nD and that it is indeed divergence-free in any dimension.

As we have also seen, 2D and 3D divergence free vector noise can be formulated both in terms of curl as Bridson [2007] did, as well as in terms of a cross product as DeWolf [2005] and we do, but the formulations are not equivalent. Some curl vector fields (in 3D) cannot be generated as the cross product of two gradient vector fields.

While this can be seen as a limitation of our method in the sense that we loose some expressivity, we get significant benefits in return for constraining the integral curves of the noise vector field to be intersections of iso-contours. Besides the fact that the cross product formulation generalizes to arbitrary dimensions, the proposed reprojection scheme provides much better precision and, in some cases, better trade-offs between performance and precision. For instance, we note from Table 2 how 64 steps of RK4 with reprojection beats 512 steps of Euler integration both in terms of performance and RMSE error. Moreover, in practice, 3D curl noise vector fields are constructed by computing the curl of a vector field obtained by conjoining three independent noise functions. Theoretically this is more expressive, but we are unsure how to exploit this expressivity. Finally, in 2D our method has the same expressivity as curl noise – while being more precise thanks to reprojection.

6.1 Future Work

Our nD curl-noise formulation opens up a wide range of unexplored possibilities, including new ways to define projection operators in 2D, 3D, and higher dimensions. This article merely scratches the surface. For instance, one could induce anisotropy in 3D via a (3+k)D projection operator (with $k > 0$), analogous to our 2D example of Section 5.5. We have also demonstrated that using our DFVN fields to jitter points in arbitrary dimensions is possible. One of the most apparent next steps is to investigate the application of this method to concrete problems such as integrating high-dimensional functions or sampling problems.

Acknowledgments

We wish to thank the LORIA laboratory for supporting our collaboration and Jens Gravesen for elucidating aspects of the exterior derivative. This project has received funding from Innovation Fund Denmark (0223-00041B).

References

J Andreas Bærentzen, Jeppe Revall Frisvad, and Jonàs Martínez. 2023. Curl noise jittering. In *SIGGRAPH Asia 2023 Conference Papers*. ACM, 88:1–88:11. doi:10.1145/3610548.3618163

Samir Brahim Belhaouari, Yunis Carreon Kahalan, Ilyasse Aksikas, Abdelouahed Hamdi, Ismael Belhaouari, Elias Nabel Haoudi, and Halima Bensmail. 2025. Generalizing the cross product to N dimensions: A novel approach for multidimensional analysis and applications. *Mathematics* 13, 3 (2025), Article 514. doi:10.3390/math13030514

Robert Bridson. 2007. Fast Poisson disk sampling in arbitrary dimensions. In *SIGGRAPH 2007 Sketches*. ACM, 22. doi:10.1145/1278780.1278807

Robert Bridson. 2015. *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press. doi:10.1201/b10635

Robert Bridson, Jim Houriham, and Marcus Nordenstam. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics* 26, 3 (2007), 46:1–46:3. doi:10.1145/1275808.1276435

Eric Brochu, Tyson Brochu, and Nando De Freitas. 2010. A Bayesian interactive optimization approach to procedural animation design. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '10)*.

Tyson Brochu and Robert Bridson. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing* 31, 4 (2009). doi:10.1137/080737617

John Charles Butcher. 1996. A history of Runge-Kutta methods. *Applied numerical mathematics* 20, 3 (1996), 247–260. doi:10.1016/0168-9274(95)00108-5

Jumyung Chang, Ruben Partono, Vinicius C. Azevedo, and Christopher Batty. 2022. Curl-Flow: boundary-respecting pointwise incompressible velocity interpolation for grid-based fluids. *ACM Transactions on Graphics* 41, 6 (2022), 243:1–243:21. doi:10.1145/3550454.3555498

Qiang Chen, Tingsong Lu, Yang Tong, Guoliang Luo, Xiaogang Jin, and Zhigang Deng. 2022. A practical model for realistic butterfly flight simulation. *ACM Transactions on Graphics* 41, 3 (2022). doi:10.1145/3510459

Ivan DeWolf. 2005. Divergence-free noise. https://www.academia.edu/download/50013242/DFnoiseR.pdf

Xinwen Ding and Christopher Batty. 2023. Differentiable curl-noise: boundary-respecting procedural incompressible flows without discontinuities. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 1 (2023), 16:1–16:16. doi:10.1145/3585511

Tevian Dray. 1999. The hodge dual operator. *Oregon State University report* (1999), 1–6.

Sevinc Eroglu, Sascha Gebhardt, Patric Schmitz, Dominik Rausch, and Torsten Wolfgang Kuhlen. 2018. Fluid sketching - Immersive sketching based on fluid flow. In *Virtual Reality and 3D User Interfaces (VR)*. IEEE, 475–482. doi:10.1109/vr.2018.8446595

Stefan Gustavson. 2005. Simplex noise demystified. https://itn-web.it.liu.se/~stegu76/simplexnoise/simplexnoise.pdf

Christopher Horvath and Willi Geiger. 2009. Directable, high-resolution simulation of fire on the GPU. *ACM Transactions on Graphics* 28, 3 (2009), 41:1–41:8. doi:10.1145/1576246.1531347

Theodore Kim, Nils Thürey, Doug James, and Markus Gross. 2008. Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics* 27, 3 (2008), 50:1–50:6. doi:10.1145/1399504.1360649

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. *Computer Graphics Forum* 29, 8 (2010), 2579–2600. doi:10.1111/j.1467-8659.2010.01827.x

Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Transactions on Graphics* 27, 5 (2008), 166:1–166:8. doi:10.1145/1457515.1409119

Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive localized liquid motion editing. *ACM Transactions on Graphics* 32, 6 (2013), 184:1–184:10. doi:10.1145/2508363.2508429

Axel Paris, Adrien Peytavie, Eric Guérin, Oscar Argudo, and Eric Galin. 2019. Desertscape simulation. *Computer Graphics Forum* 38, 7 (2019), 47–55. doi:10.1111/cgf.13815

Ken Perlin. 1985. An image synthesizer. *Computer Graphics (SIGGRAPH '85)* 19, 3 (1985), 287–296. doi:10.1145/325165.325247

Ken Perlin. 2002. Noise Hardware. In *Real-Time Shading Languages*. Chapter 2, SIGGRAPH 2002 Course Notes, Article 36. https://www.csee.umbc.edu/~olano/s2002c36/ch02.pdf

Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics* 31, 4 (2012), 112:1–112:8. doi:10.1145/2185520.2335463

Hagit Schechter and Robert Bridson. 2008. Evolving sub-grid turbulence for smoke animation. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*.

Ronald Shaw. 1987. Vector cross products in n dimensions. *International Journal of Mathematical Education in Science and Technology* 18, 6 (1987), 803–816. doi:10.1080/0020739870180606

Wolfram Von Funck, Holger Theisel, and Hans-Peter Seidel. 2006. Vector field based shape deformations. *ACM Transactions on Graphics* 25, 3 (2006), 1118–1125. doi:10.1145/1179352.1142002

Xinjie Wang, Xiaogang Jin, Zhigang Deng, and Linling Zhou. 2014a. Inherent noise-aware insect swarm simulation. *Computer Graphics Forum* 33, 6 (2014), 51–62. doi:10.1111/cgf.12277

Xinjie Wang, Linling Zhou, Zhigang Deng, and Xiaogang Jin. 2014b. Flock morphing animation. *Computer Animation and Virtual Worlds* 25, 3-4 (2014), 351–360. doi:10.1002/cav.1580

Yuwen Wu. 2021. Bitangent noise. https://github.com/atyuwen/bitangent_noise

Bowen Yang, William Corse, Jiecong Lu, Joshuah Wolper, and Chen-Fanfu Jiang. 2019. Real-time fluid simulation on the surface of a sphere. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2, 1 (2019), 4:1–4:17. doi:10.1145/3320285