Divergence-Free Noise

Ivan DeWolf*
Martian Labs

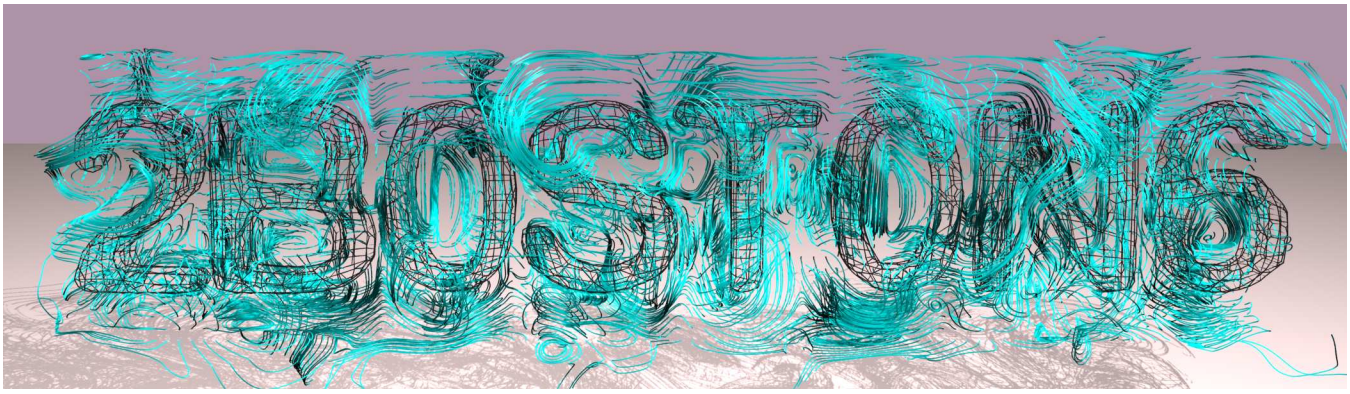

Figure 1: streamers advected by a Divergence-Free noise function conforming to complex geometry

Abstract

In this paper, we describe a method for generating a divergence-free vector field from a scalar noise function. Noise functions have numerous uses in the visual simulation of various phenomena, however they lack many of the desireable features of fluid dynamics systems. Computational fluid dynamics can accurateley model the dynamics of fluids, but generally requires data heavy dynamic simulations and significant computational expense. We present a method of generating a vector field similar to a noise function, but with some of the desireable features of the vector fields generated with computational fluid dynamics. Our method is a light, fast, analytic solver that is simple to implement in existing animation systems. This paper describes our method for constructing a divergence-free vector field, and presents results demonstrating some of the effects that can be realized with it.

Keywords: noise functions, CFD, divergence-free, vector fields

1 Introduction

Vector fields generated by noise functions have many uses in computer animation. Since their introduction in [Perlin 1985], noise functions have been used to affect particle systems, distort geometry, warp texture coordinates, and in numerous other ways to get realistic appearances in images and motion. Vector noise is a computationally light, convenient method to generate varying vector fields. However, when simulating the behavior of fluids, vector noise lacks the verisimilitude of computational fluid dynamics methods.

*e-mail: ivan@martian-labs.com

In the past several years, many techniques to accurately compute fluid motion using techniques gleaned from the computational fluid dynamics literature have been presented to the computer graphics community. Various Eulerian, Lagrangian, and hybrid simulation systems have been presented and implemented. These are all simulation systems involving the storage and retrieval of data stored in meshes or sparse particles. The frequencies these systems can resolve are dependent upon the density of the data stored, and the computational expense.

Fluid solvers are usually used to simulate incompressible flows by enforcing a divergence-free criterion upon the resulting vector field [Fedkiw et al. 2001]. Enforcing a divergence-free criterion makes the resulting vector field both volume preserving and vorticular. This results in a realistic and visually pleasing result. This criterion is often enforced through the use of computationally expensive linear solvers computing the gradient of the divergence, and then subtracting the divergence from the vector field [Tong et al. 2003].

In this paper we present a technique for directly analyticly generating a divergence-free vector field (DFV) from a scalar function. Our method has many of the desireable features of both a vector noise function and a computational fluid dynamics solution, with a light memory footprint and computational cost, and no additional cost to resolve high frequencies, while still preserving a divergence-free criterion. We also present a method to make the resulting vector field conform to bondaries defined by geometric primitives. Our method can be computed at any frequency without the need for precomputation or table storage.

2 Previous Work

Noise functions have a long history in the field of computer graphics. First developed by Ken Perlin [Perlin 1985; Perlin 2002], numerous other methods for generating solid textures have since been presented. An excellent survey of solid textures with the presentation of a sparse convolution method is presented in [Lewis 1989], a non-smooth solid texture is given in [Worley 1996], and a bandwidth limited wavelet based method is given in [Cook and

DeRose 2005]. Most implementations of solid noise are fast and light enough to be generated on a per-sample basis at rendertime, even in realtime systems. It is desireable to have a function that is as computationally light and fast as these noise functions, but with the visual appearance of turbulent fluid flows.

Generating divergence-free vector fields using a fast fourier transform and a Kolmogorov spectrum was first introduced to the graphics community in [Stam and Fiume 1993]. This technique was used to generate fire [Lamorlette and Foster 2002] and texture for large scale explosions [Rasmussen et al. 2003]. Fourier-based techniques require a grid of data to store, and some computation at every frame to fill it. The frequencies that they can resolve are a function of the grid size and resolution, which also determines the computational cost and ram usage. Fourier techniques make seamlessly tileable solutions, however they cannot handle boundary conditions. Noise functions can resolve very high frequencies in a large non-periodic domain with minimal ram usage and computational expense, since their values can be computed locally as needed. Combining a distance function with the noise function allows our method to handle boundary conditions generated by common geometric primitives.

There have been numerous advances in the field of fluid dynamics for computer graphics over the years. In recent years, direct implementations of CFD have shown realistic results. [Stam 1999] presents a method using semi Lagrangian techniques to ensure numerical stability for any size timestep. Various techniques to add vorticity back into the system to account for numerical dissipation have appeared [Fedkiw et al. 2001; Selle et al. 2005]. [Stam 2003] presents techniques to compute fluid flows on 2d surfaces in 3 dimensions. [Feldman et al. 2003] presents a technique to handle compressible flows by maintaining a local divergence value. [Tong et al. 2003] presents a method for computing Hodge decompositions on tetrahedral meshes. Flows on hybrid meshes [Feldman et al. 2005] have been presented. There have been several papers on fully Lagrangian techniques, such as [Clavet et al. 2005], smoothed particle hydrodynamics [Desbrun and Cani 1996], and vortex filaments [Angelidis and Neyret 2005]. Our solution is not a fluid dynamics solver, since it does not inherently handle the energy conservation and advection that a true CFD solution has. However, the features that it does share combined with its light computational footprint makes it a useful tool in the visual simulation of fluid phenomena.

3 Constructing Divergence-Free Vector Fields

Our method for directly constructing a divergence-free vector field takes two scalar functions and combines their outputs to generate a vector field. A Helmholtz-Hodge decomposition will split any smooth vector field into three components; a divergent vector field, a curl vector field, and a harmonic vector field. A divergence free vector field (or *solenoidal* field) can contain any combination of curl and harmonic vector fields, but no divergence. Similarly, a curl-free vector field (or *irrotational* field) can contain any combination of harmonic and divergent elements, but no curl. The gradient of any scalar field is a curl free vector field. Typicly, a divergence free vector field is created by computing and then removing the divergent component from an input vector field. While there are many good techniques for performing this computation (an excellently described method is [Tong et al. 2003]), they usually involve the use of data grids and computational expense. We bypass this overhead by directly deriving a divergence free vector field as follows:

Given two vector fields U and V where

$$\nabla \times U = \nabla \times V = 0$$

$$\nabla \cdot (U \times V) = 0 \tag{1}$$

Given two scalar fields, ϕ and ψ, we can generate a DFV

$$\nabla \cdot ((\nabla \phi) \times (\nabla \psi)) = 0 \tag{2}$$

These two scalar fields can be generated by two noise functions with different random seeds. The gradient of a scalar function defines the normals to the isosurfaces in the scalar field. The vector field we generate with our method follows the isocontours where the two isosurfaces intersect. On any surface in three dimensions, we can generate a DFV as the cross product of the gradient of a noise function (tangent to the surface, in 2d) and the surface normal.

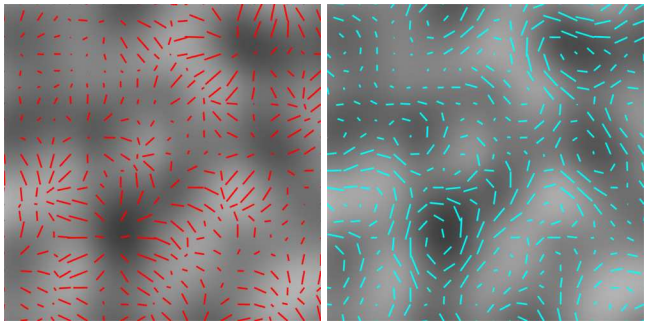

Figure 2: in the 2-d case, cross the gradient with the surface normal

4 Conforming to Boundaries

Our method for generating a DFV computes vectors from two scalar fields. In a scalar field generated as a function of the distance to a geometric primitive, the isocontours are parallel to the surface and obviously do not cross the boundary. Combining the distance field with a noise function gives us a scalar field from which we can construct our DFV. This DFV will not cross the boundaries defined by the geometry, and will flow around it in a realistic manner.

$$\nabla \cdot ((\nabla (\phi * \delta)) \times (\nabla \psi)) = 0 \tag{3}$$

Where ϕ and ψ are noise functions with different seeds, and δ is a distance function. The distance function can be clipped, making the surface only affect flows near it, by not contributing to the slope of the noise function a certain distance away from the surface.

5 Future Work

The two incompressible Euler equations state that the velocity should conserve both mass and momentum; using our method directly derives a vector field that will conserve mass, but not necessarily conserve momentum. The total momentum at a given time step is the integral of the slopes of the two scalar fields used; the energy spectrum and conservation depends on the methods used to

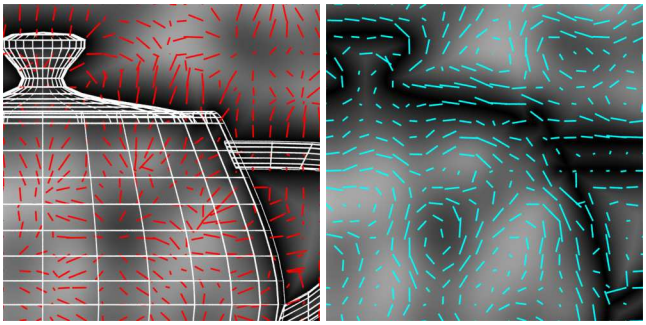

Figure 3: combining the distance function gives a DFV that conforms to geometry

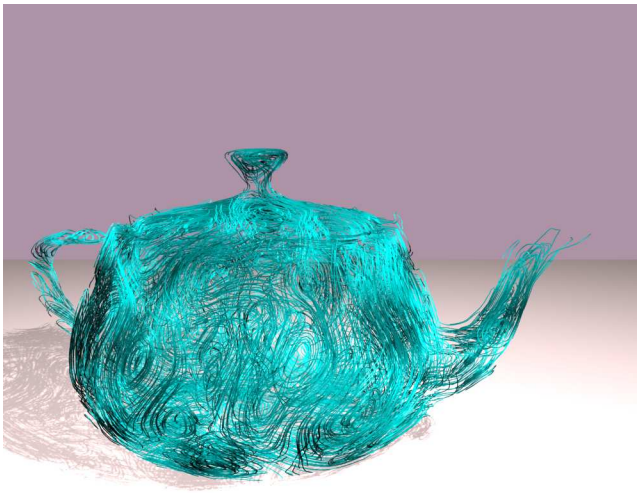

Figure 4: streamers advected by a 3d constrained DFV

generate these scalar fields. This is an area ripe for further exploration. Noise functions are often animated by by using a four dimensional noise function and offsetting the sampled location in the fourth dimension. Although this method gives visually acceptable results, it changes the slopes randomly which adds and absorbs momentum from the system. Most of our examples use this method.

One method of ensuring momentum conservation is to normalize the resulting vector field, however normalizing a divergence-free vector field will introduce some divergence into the system. For some applications this may be sufficient, trading off accuracy in mass conservation for the apparent conservation of momentum.

Another method of animating a DFV is to use static scalar fields and distort them to achieve some animation. Since the momentum is a function of the slope, the deformations cannot alter the slope of the initial scalar fields to conserve momentum. Using a DFV to distort a scalar field will cause the slopes to translate and rotate, but without divergence they will not stretch or squash. Advecting the scalar fields by the vector fields they generate would be a more accurate interpretation of the incompressible Euler equations, but would be difficult to achieve without large storage grids.

6 Implementation Issues

In our implementation, we compute the gradient of the noise function by sampling it at the current location and at offset locations, and use the deltas to compute the gradient. This is a good general purpose gradient computation, and will work with any smoothly varying scalar functions, including most implementations of solid textures. However, it does require the knowledge of the frequency of the details in the scalar function, and it takes several calls to the scalar function to calculate. Many types of noise functions have faster ways to directly access their gradients, which would significantly improve the performance of the algorithm.

Most algorithms to determine the closest point on a group of geometric primitives have a fairly high computational expense. For many of our examples, we used a 3-d texture file filled with scalar distance values and then simply looked up values in them. Faster methods to determine distance to geometry would be useful for this.

Although the vector field generated is divergence-free, most integration schemes will introduce divergence into the system. This is especially pronounced in areas of high curvature (e.g. vortex centers). Using a simple Eulerian ODE solver is fast, and for small timesteps, it does work reasonably well (It is the method we used for all our examples). However, to get greater accuracy for simulations of a long duration, a degree 4 Runge-Kutta solver with adaptive timesteps works well, but with considerable computational overhead. In our tests, the Eulerian method was accurate enough to not require the added computatuion of the Runge-Kutta solver.

7 Results and Discussion

Using our method to generate a 2d DFV on a 3d surface, we created a particle system simulating weather-like patterns on a globe (figure 6). We added a sine wave in the Y direction to the noise function, wich caused the jetstream-like winds at the poles and coriolis winds at the equator.

Using our system we created soap bubbles, with the swirling fluids being generated entireley in the shader during rendering (figure 5). The shader does not use any external data or custom DSO's, it was written entireley in the existing shading language.

References

ANGELIDIS, A., AND NEYRET, F. 2005. Simulation of smoke based on vortex filament primitives. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005*, ACM Press / ACM SIGGRAPH, ACM, 87–96.

CLAVET, S., BEAUDOIN, P., AND POULIN, P. 2005. Particle-based viscoelastic fluid simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005*, ACM Press / ACM SIGGRAPH, D. Terzopoulos and V. Zordan, Eds., ACM, 219–228.

COOK, R. L., AND DEROSE, T. 2005. Wavelet noise. In *Proceedings of SIGGRAPH 2005*, ACM Press / ACM SIGGRAPH, M. Gross, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 803–811.

DESBRUN, M., AND CANI, M.-P. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In

Eurographics Workshop on Computer Animation and Simulation (EGCAS), Springer-Verlag, R. Boulic and G. Hegron, Eds., ACM, 61–76. Published under the name Marie-Paule Gascuel.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND ARIKAN, O. 2003. Suspended particle explosions. In *Proceedings of SIGGRAPH 2003*, ACM Press / ACM SIGGRAPH, J. K. Hodgins, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 708–715.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGER, B. M. 2005. Animating gasses with hybrid meshes. In *Proceedings of SIGGRAPH 2005*, ACM Press / ACM SIGGRAPH, M. Gross, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 904–909.

LAMORLETTE, A., AND FOSTER, N. 2002. Structural modeling of flames for a production environment. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, ACM, 729–735.

LEWIS, J. P. 1989. Algorithms for solid noise synthesis. In *Proceedings of SIGGRAPH 1989*, ACM Press / ACM SIGGRAPH, J. Lane, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 263–270.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23*, 3, 457–462.

NGUYEN, D. Q., FEDKIW, R., AND JENSEN, H. W. 2002. Physically based modeling and animation of fire. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, ACM, 721–728.

PERLIN, K. 1985. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 287–296.

PERLIN, K. 2002. Improving noise. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, J. F. Hughes, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 681–682.

RASMUSSEN, N., NGUYEN, D. Q., GEIGER, W., AND FEDKIW, R. 2003. Smoke simulation for large scale phenomena. *ACM Trans. Graph. 22*, 3, 703–707.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. In *Proceedings of SIGGRAPH 2005*, ACM Press / ACM SIGGRAPH, M. Gross, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 910–914.

STAM, J., AND FIUME, E. 1993. Turbulent wind fields for gaseous phenomena. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, ACM, 369–376.

STAM, J. 1999. Stable fluids. In *Proceedings of SIGGRAPH 1999*, ACM Press / ACM SIGGRAPH, A. Rockwood, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 121–128.

STAM, J. 2003. Flows on surfaces of arbitrary topology. In *Proceedings of SIGGRAPH 2003*, ACM Press / ACM SIGGRAPH, J. K. Hodgins, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 724–731.

TONG, Y., LOMBEYDA, S., HIRANI, A. N., AND DESBRUN, M. 2003. Discrete multiscale vector field decomposition. In *Proceedings of SIGGRAPH 2003*, ACM Press / ACM SIGGRAPH, J. K. Hodgins, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 445–452.

WORLEY, S. 1996. A cellular texture basis function. In *Proceedings of SIGGRAPH 1996*, ACM Press / ACM SIGGRAPH, H. Rushmeier, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 291–294.

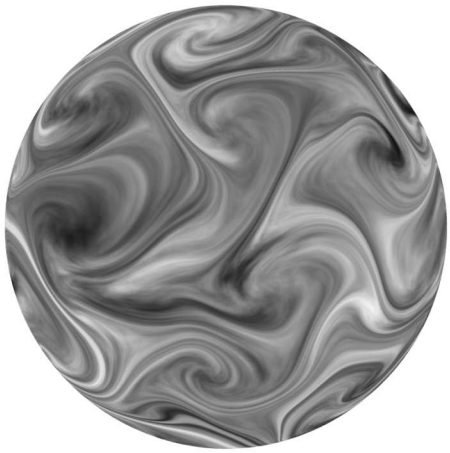

Figure 5: divergence-free noise computed in a surface shader

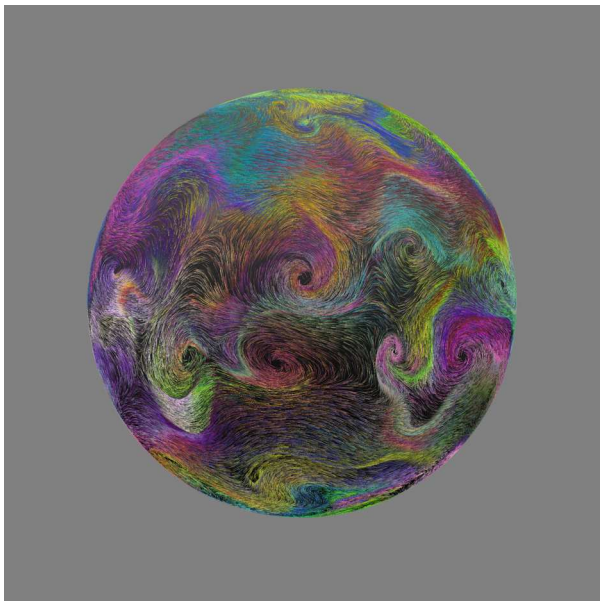

Figure 6: global weather patterns with jetstreams