Convex Optimization for Guided Fluid Simulation

Owen Jow
UC San Diego
owen@eng.ucsd.edu

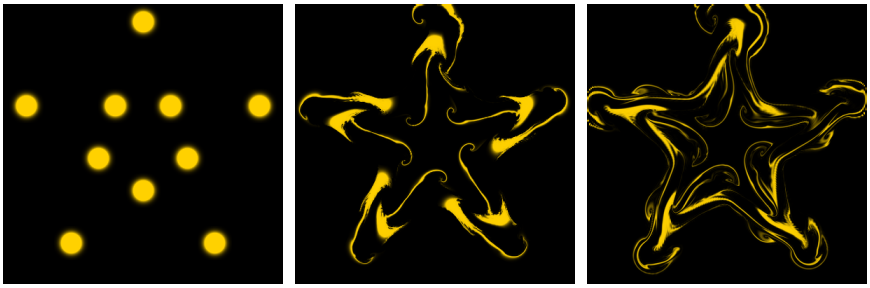

Figure 1: Initial, middle, and late stages of a guided five-point star simulation

Abstract

In this project, I formulate guided Eulerian fluid simulation as a convex optimization problem and make a comparison between sparse least-squares and a proximal algorithm as methods for solving it. Ultimately, I find that I am able to successfully realize a guided simulation while maintaining qualities of realistic fluid motion, where the balance between guided and natural motion is controlled by a hyperparameter. I conclude that optimization provides an effective perspective for simultaneously achieving both natural fluid simulation and artificial fluid guiding.

1 Introduction

Fluid simulation and convex optimization are both well-studied subjects with many applications. Here I would like to use one in the context of the other; namely, I would like to use convex optimization to perform not just fluid simulation, but *guided* fluid simulation. The setup is simple: given an initial state and a target velocity field, I want to stay close to the target velocity field while also exhibiting the kind of motion that is reminiscent of a real incompressible fluid.

In short, there are two objectives: one, respect the laws of fluid motion (the Navier-Stokes equations), and two, maintain a guided flow by following the target velocity.

1.1 Motivation

The tendency of humanity is to control, enslave, redirect, and remake. It is not enough to observe our environment; we must bend it to our will. We do this not only physically but virtually too. Can an artist or animator be wholly satisfied with depicting a fluid as it is, rather than as it could be?

That being said, it is often difficult to realistically achieve targeted behavior. Many existing techniques are either inefficient or ineffective in certain cases. For example, the easiest programmatic solution – having an artist supply all of the forces in a vanilla interactive simulation – requires an incredible amount of re-simulation and tuning over a series of time steps. The ideal solution is one which can be performed near-automatically by computers, without any human input except for (1) a specification of the desired velocity field, (2) hyperparameter settings, and (3) perhaps some touch-ups at the end.

Note that the difficulty of the problem stems from the fact that it has two objectives: *realism* and *guidance*. It is easy enough to achieve each on its own, but challenging to satisfactorily balance them both at the same time. A big motivation for using an optimization framework is that it provides an obvious way to handle multiple goals and to control the tradeoff between each of them: just formulate a problem with multiple weighted terms in the objective. Then, once the problem is set up, there are many different approaches within the optimization world for solving it.

The best part is that in this case, we are able to formulate the problem as a *convex* optimization problem (see section 2.1), making it even easier to solve (as local minima are also global minima).

1.2 Background

Eulerian fluid simulation revolves around solving the incompressible Navier-Stokes equations:

$$\frac{\partial \mathbf{v}}{\partial t} = -\mathbf{v} \cdot \nabla \mathbf{v} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{v} + \mathbf{f}$$
$$\nabla \cdot \mathbf{v} = 0$$

with $\mathbf{v}$ being velocity, ρ being density, p being pressure, ν being kinematic viscosity, and $\mathbf{f}$ being acceleration due to external forces. Basically, the first equation says that the acceleration of the fluid at each fixed point in the grid is equal to the sum of advection, pressure, viscosity, and external force (e.g. gravity) terms. The second equation says that the velocity field should be divergence-free.

While I could spend a lot of time describing basic fluid simulation, that is not really the focus of this project. For more details on interpreting and numerically solving Navier-Stokes, I refer readers to the early papers on graphics-oriented fluid simulation ([1], [2], [3]). Note that since the 90s when those papers were released, fluid simulation has experienced a significant amount of further development.

People have traditionally achieved guided, grid-based fluid simulation in different ways. One option is to utilize driving forces and gathering terms as per [4], but the resulting flows are not always realistic. [5] uses gradient descent for guiding, but is slow due to having to a run an entire simulation at each optimization step. [6] employs a variational approach which effectively upsamples an artist's low-resolution guidance, but still requires the artist to tune the simulation in low resolution and struggles with inter-resolution differences. [7] introduces the general proximal method that I emulate.

1.3 Contributions

I have implemented a guided 2D fluid simulation program from an optimization perspective.

2 Problem Statement

2.1 Primal Formulation

I follow the setup from [7]:

$$\begin{aligned} \underset{\mathbf{v}}{\text{minimize}} \quad & \|\mathbf{G}(\mathbf{v} - \mathbf{v}_{\text{target}})\|_2^2 + \|\mathbf{W}(\mathbf{v} - \mathbf{v}_{\text{current}})\|_2^2 \\ \text{subject to} \quad & \nabla \cdot \mathbf{v} = \mathbf{0} \end{aligned}$$

where $\mathbf{v}_{\text{target}}$ is the (user-specified) target velocity field, $\mathbf{v}_{\text{current}}$ is the current velocity field, $\mathbf{G}$ is a Gaussian blur matrix, and $\mathbf{W}$ is a diagonal matrix of weights describing the resistance to guiding at each location. $\mathbf{v}$, $\mathbf{v}_{\text{target}}$, and $\mathbf{v}_{\text{current}}$ are each vectors $\in \mathbb{R}^{2wh}$ (where w is width and h is height) representing flattened 2D velocity fields. $\mathbf{G}$ and $\mathbf{W}$ are each sparse matrices $\in \mathbb{R}^{2wh \times 2wh}$.

Next, note that via finite differencing you can approximate the divergence at location (i, j) as $\mathbf{d}^\top \mathbf{u}$ where $\mathbf{d} = \begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}^\top$ and $\mathbf{u} = \begin{bmatrix} v_{x_{i+1j}} & v_{x_{i-1j}} & v_{y_{ij+1}} & v_{y_{ij-1}} \end{bmatrix}^\top$.

Thus the primal problem can be rewritten as

$$\begin{aligned} \underset{\mathbf{v}}{\text{minimize}} \quad & \|\mathbf{G}(\mathbf{v} - \mathbf{v}_{\text{target}})\|_2^2 + \|\mathbf{W}(\mathbf{v} - \mathbf{v}_{\text{current}})\|_2^2 \\ \text{subject to} \quad & \mathbf{D}\mathbf{v} = \mathbf{0} \end{aligned}$$

where $\mathbf{D}$ is a sparse matrix containing a row for each 2D grid location, with the entries of $\mathbf{d}$ in the appropriate slots. Finally, expanding the primal objective function, we obtain

$$\begin{aligned}
\underset{\mathbf{v}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{v}^\top \mathbf{A}\mathbf{v} + \mathbf{b}^\top \mathbf{v} + c \\
\text{subject to} \quad & \mathbf{D}\mathbf{v} = \mathbf{0} \\
& \mathbf{A} = 2\mathbf{G}^\top \mathbf{G} + 2\mathbf{W}^2 \\
& \mathbf{b} = -2\mathbf{G}^\top \mathbf{G}\mathbf{v}_{\text{target}} - 2\mathbf{W}^2 \mathbf{v}_{\text{current}} \\
& c = \mathbf{v}_{\text{target}}^\top \mathbf{G}^\top \mathbf{G}\mathbf{v}_{\text{target}} + \mathbf{v}_{\text{current}}^\top \mathbf{W}^2 \mathbf{v}_{\text{current}}
\end{aligned}$$

The objective function, being a standard-form quadratic, is clearly convex, and the feasible set of divergence-free velocity fields is also convex (as per [8]). So the problem as a whole is convex.

2.2 Dual Formulation

The Lagrange dual problem (using the variables $\mathbf{A}$, $\mathbf{b}$, and c from before) is

$$\underset{\nu}{\text{maximize}} \quad \inf_{\mathbf{v}} \left(\frac{1}{2}\mathbf{v}^\top \mathbf{A}\mathbf{v} + (\mathbf{b} + \mathbf{D}^\top \nu)^\top \mathbf{v} + c \right)$$

When I evaluate the infimum in the Lagrange dual function by taking the gradient and setting it to 0, I find that $\mathbf{v} = -\mathbf{A}^{-1}(\mathbf{b} + \mathbf{D}^\top \nu)$. Substituting this back into the Lagrange dual problem, I obtain

$$\begin{aligned}
& \underset{\nu}{\text{maximize}} \quad \frac{1}{2}\left[\mathbf{A}^{-1}(\mathbf{b} + \mathbf{D}^\top \nu)\right]^\top (\mathbf{b} + \mathbf{D}^\top \nu) - (\mathbf{b} + \mathbf{D}^\top \nu)^\top \mathbf{A}^{-1}(\mathbf{b} + \mathbf{D}^\top \nu) + c \\
\implies \quad & \underset{\nu}{\text{maximize}} \quad -\frac{1}{2}(\mathbf{b} + \mathbf{D}^\top \nu)^\top \mathbf{A}^{-1}(\mathbf{b} + \mathbf{D}^\top \nu) + c
\end{aligned}$$

2.3 KKT Conditions

Again, I use the variables $\mathbf{A}$, $\mathbf{b}$, and c as defined in the primal.

$$\begin{aligned}
\mathbf{D}^\top \mathbf{v}^* &= \mathbf{0} \\
\mathbf{A}\mathbf{v}^* + \mathbf{b} + \mathbf{D}^\top \nu^* &= \mathbf{0}
\end{aligned}$$

3 Approach

At each time step, I use standard physical simulation techniques to create the “current" velocity field at each frame, and then solve the optimization problem to determine the final guided velocity.

The following subsections describe my two approaches for solving the optimization problem:

3.1 Least-Squares

Since the objective function $f(\mathbf{v})$ is convex, its minimum is attained at the $\mathbf{v}$ for which $\nabla f(\mathbf{v}) = \mathbf{0}$:

$$\begin{aligned}
\nabla \left(\frac{1}{2}\mathbf{v}^\top \mathbf{A}\mathbf{v} + \mathbf{b}^\top \mathbf{v} + c \right) &= \mathbf{0} \\
\mathbf{A}\mathbf{v} + \mathbf{b} &= \mathbf{0} \\
\mathbf{A}\mathbf{v} &= -\mathbf{b}
\end{aligned}$$

There is also the $\mathbf{D}\mathbf{v} = \mathbf{0}$ constraint, making this a sparse least-squares problem $\begin{bmatrix} \mathbf{A} \\ \mathbf{D} \end{bmatrix} \mathbf{v} = \begin{bmatrix} -\mathbf{b} \\ \mathbf{0} \end{bmatrix}$.
Thus I can solve it using `SciPy`'s sparse LSQR solver.

3.2 Proximal Algorithm

I can also apply a proximal method ([9]), i.e. a general, efficient algorithm that can be used for solving constrained, non-smooth problems at scale.

Following [7], [9], and [10], the proximal operator of the quadratic objective function f is

$$\begin{aligned}
\text{prox}_f(\mathbf{q}) &= \arg\min_{\mathbf{v}} \left(f(\mathbf{v}) + \frac{1}{2}\|\mathbf{v}-\mathbf{q}\|^2 \right) \\
&= \arg\min_{\mathbf{v}} \left(\frac{1}{2}\mathbf{v}^\top \mathbf{A}\mathbf{v} + \mathbf{b}^\top \mathbf{v} + c + \frac{1}{2}\|\mathbf{v}-\mathbf{q}\|^2 \right) \\
&= (\mathbf{A}+\mathbf{I})^{-1}(\mathbf{q}-\mathbf{b}) \\
&= \mathbf{v}_{\text{current}} + (2\mathbf{G}^\top\mathbf{G} + 2\mathbf{W}^2 + \mathbf{I})^{-1}[\mathbf{q} + 2\mathbf{G}^\top\mathbf{G}(\mathbf{v}_{\text{target}} - \mathbf{v}_{\text{current}}) - \mathbf{v}_{\text{current}}]
\end{aligned}$$

I can use this in the primal-dual proximal method of [7] and [10] to solve the optimization problem by maintaining three extra variables, $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ (each with the same dimensionality as the velocity field) and applying the following updates for some number of iterations during each fluid step:

$$\begin{aligned}
\mathbf{x} &= \mathbf{x} + \mathbf{y} - \text{prox}_f(\mathbf{x}+\mathbf{y}) \\
\mathbf{v} &= \text{prox}_g(\mathbf{z}-\mathbf{x}) \\
\mathbf{y} &= 2\mathbf{v} - \mathbf{z} \\
\mathbf{z} &= \mathbf{v}
\end{aligned}$$

where prox_g can be computed as a pressure projection as per [11].

4 Results

Description	URL
Initial velocity only, no per-step guiding	`https://gfycat.com/bouncytinyamurratsnake`
Per-step guiding with sparse least-squares	`https://gfycat.com/hastyordinarycheetah`
Per-step guiding with the proximal method	`https://gfycat.com/spitefulliquidambushbug`

Table 1: Links to fluid simulation GIFs

In the above GIF examples, I attempt to guide the fluid to adopt and maintain a five-point star velocity. As you can see from the first example, without per-step guiding the tendency of the fluid is to spread out in a somewhat unpredictable way, even when the velocity is initialized to the target velocity. I would argue that both the LSQR and proximal methods are successful at creating fluid-like guided velocity fields (for example, they retain some of the "mushroom" effects from pressure projection). In my implementation, the LSQR method is faster: for a 400×400 grid and identical initial and target conditions, the LSQR method takes $\sim$0.6s per update while the proximal method takes $\sim$4.0s.

5 Conclusion and Future Work

I have created a 2D fluid simulation which can be guided using convex optimization. Since the optimization problem is an equality-constrained quadratic minimization, it can be solved relatively easily, flexibly, and satisfactorily, as I have heretofore demonstrated with both of the methods I tried.

There are many potential extensions to this project. One could enhance the simulation in ways that don't relate to optimization (e.g. 3D, better rendering, support for more types of fluids), experiment with other iterative solvers (e.g. conjugate gradient, ADMM), or use optimization with deep learning[1].

[1] e.g. by optimizing a network to map (current velocity, target velocity) onto the space of realistic (current velocity, next velocity) pairs while also minimizing the difference between target velocity and next velocity

References

[1] Nick Foster and Dimitris Metaxas. Realistic animation of liquids. *CVGIP: Graphical Model and Image Processing*, 58:471–483, 01 1996.

[2] Nick Foster and Dimitris Metaxas. Modeling the motion of a hot, turbulent gas. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 181–188, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[3] Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 121–128, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[4] Raanan Fattal and Dani Lischinski. Target-driven smoke animation. *ACM Trans. Graph.*, 23(3):441–448, August 2004.

[5] Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. Keyframe control of smoke simulations. *ACM Trans. Graph.*, 22(3):716–723, July 2003.

[6] Michael Nielsen, Brian Christensen, Nafees Zafar, Doug Roble, and Ken Museth. Guiding of smoke animations through variational coupling of simulations at different resolution. pages 217–226, 08 2009.

[7] Tiffany Inglis, Marie-Lena Eckert, James Gregson, and Nils Thürey. Primal-dual optimization for fluids. *CoRR*, abs/1611.03677, 2016.

[8] P. Y. Simard and G. E. Mailloux. A projection operator for the restoration of divergence-free vector fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(2):248–256, March 1988.

[9] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014.

[10] Thomas Pock, Daniel Cremers, Horst Bischof, and Antonin Chambolle. An algorithm for minimizing the mumford-shah functional. pages 1133–1140, 11 2009.

[11] James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. From capture to simulation: Connecting forward and inverse problems in fluids. *ACM Trans. Graph.*, 33(4), July 2014.