Smoke Simulation Using Fourier Spectral Method

DIYANG ZHANG, McGill University

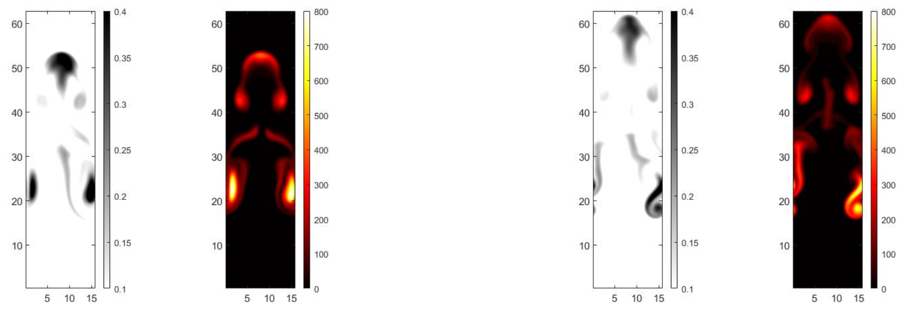

Fig. 1. Density (in white) and temperature (in black) plot of cross section of 3D smoke simulated using our Fourier spectral method. The data is generated in C and the pictures are drawn in Matlab.

Fluid animation has become one of the most popular aspects in physical simulation. It often refers to technique for emulating the realistic animation for fluid such as water or smoke. Traditionally, *advection-projection* scheme is appreciated as the common fluid solver for its efficiency. We introduced here an original numerical method under this scheme, based on Fourier spectral approximations, for simulating fire and smoke. The structure is sufficiently concise, and thus it is extremely easy to generalize. We will utilize this method to produce eventually several beautiful fluids.

1 INTRODUCTION

Fluid animation is typically focused on automatically generating detailed and physically-reasonable visual behavior of a fluid such as water, smoke and explosions. Current techniques always rely on approximate solutions to Navier-Stokes equations that govern the real fluid dynamics:

$$\frac{\partial \mathbf{u}}{\partial \mathbf{t}} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla \mathbf{p} + \mathbf{f} \tag{1}$$

$$\nabla \cdot \mathbf{u} = \mathbf{0} \tag{2}$$

We use the standard notation: $\mathbf{u}$ for fluid velocity, $\mathbf{t}$ for time, $\mathbf{p}$ for pressure and $\mathbf{f}$ denotes external forces such as buoyancy and gravity.

Advection-Projection methods are widely used to deal with these partial differential equations. Fedkiw et al. [2001] presents their methods to solve these equations in two steps: first we solve the Equation.1 without pressure term as advection step, then we force the velocity field $\mathbf{u}^*$ to be imcompressible using a projection method.

$$\frac{\mathbf{u}^* - \mathbf{u}}{\Delta t} + (u \cdot \nabla)u = \mathbf{f} \tag{3}$$

$$\nabla^2 p = \frac{1}{\Delta t}\nabla \cdot \mathbf{u}^* \tag{4}$$

Equation 4 is solved with a boundary condition $\frac{\partial \mathbf{p}}{\partial \mathbf{n}} = 0$. It is indeed equivalent to compute the Poisson equation with pure Neumann boundary condition at a boundary point with normal $\mathbf{n}$. The intermediate velocity field $\mathbf{u}$ then becomes imcompressible by subtracting the gradient of pressure from it.

$$\mathbf{u} = \mathbf{u}^* - \Delta t \nabla p \tag{5}$$

While velocity determines the movement of fluid flow animation, density ρ and temperature $\mathcal{T}$ depict the instantaneous scene at every single time step. Assuming these scalar qualities are moving along the velocity field, we construct the equations for evolution of these two important qualities.

$$\frac{\partial \mathcal{T}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathcal{T} \tag{6}$$

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho \tag{7}$$

At the meantime, the velocity is affected by both density and temperature. For example, especially for smoke, we define the following simple model to describe these effects such as buoyancy, using the $\mathbf{z}$-vector (0,0,1), positive constant α and β and ambient temperature of the air $\mathcal{T}_{amb}$.

$$\mathbf{f}_{buoy} = -\alpha\rho\mathbf{z} + \beta(\mathcal{T} - \mathcal{T}_{amb})\mathbf{z} \tag{8}$$

All the above demonstrates the general scheme of advection-projection. We will use our Fourier spectral method to solve these equations. We present how we apply our theorem to handle different type of partial differential equations in the following section, and we summarize our algorithm for smoke simulation in Section 3.

2 RELATED WORK

We start by looking at one-dimensional Poisson equation bounded by periodic circle.

$$\mathbf{u}_{xx} = f(x) \qquad x \in [0, 2\pi]$$

We can rewrite both sides in Fourier series, as $\hat{\mathbf{u}}$ and $\hat{f}$. Then by plugging into original PDE, it is direct to solve for $\hat{\mathbf{u}}$ and transfer it back to real space.

$$\mathbf{u}(x) = \sum_{k=-\infty}^{\infty} \hat{\mathbf{u}}(k)e^{ikx}$$
$$f(x) = \sum_{k=-\infty}^{\infty} \hat{f}(k)e^{ikx}$$
$$\Rightarrow \quad \sum_k (-k^2)\hat{\mathbf{u}}(k)e^{ikx} = \sum_k \hat{f}(k)e^{ikx}$$
$$\Rightarrow \quad \hat{\mathbf{u}}(k) = \frac{\hat{f}(k)}{-k^2}$$

Through identical process, we can generalize this computation to two-dimension, where we obtain $\hat{\mathbf{u}}(k_x, k_y) = -\hat{f}(k_x, k_y)/(k_x^2 + k_y^2)$. And three-dimensional results thereby follows similarly.

When we move our concentration to heat equation, as time variant is involved, numerical method is applied here instead of straight-forward approach. We derive the following from the elementary structure $\mathbf{u}_t = \mathbf{D}\mathbf{u}_{xx}$ of heat equation,

$$\frac{\hat{u}^{n+1} - \hat{u}^n}{\Delta t} = (-k^2\mathbf{D}\hat{u}^{n+1} - k^2\mathbf{D}\hat{u}^n)/2$$

And we solve for $\hat{u}^{n+1}$,

$$\hat{u}^{n+1} = (\frac{1}{\Delta t} + \mathbf{D}\frac{k^2}{2})^{-1}(\frac{1}{\Delta t} - \mathbf{D}\frac{k^2}{2})\hat{u}^n$$

In addition, if obstacles are introduced to the system, we utilize predictor-corrector method. For heat equation, we subtract an additional term on the right-hand side and derive $\mathbf{u}_t = \Delta\mathbf{u} - \frac{1}{\eta}\mathbb{1}_\Omega(\mathbf{u} - \mathbf{u}_B)$, where η is sufficiently small and $\mathbb{1}_\Omega$ is the smooth indicator function constrained by the obstacle Ω. To solve this,

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = (\Delta\mathbf{u}^* - \Delta\mathbf{u}^n)/2 \tag{9}$$
$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\eta}\mathbb{1}_\Omega(\mathbf{u} - \mathbf{u}_B) \tag{10}$$

By separating the calculation into two different steps, we first compute the intermediate value $\mathbf{u}^*$, and thereafter we solve Equation 10 to correct our solution with the obstacle involved.

3 IMPLEMENTATION

We strictly follow the advection-projection scheme that we have mentioned in Section 1. For each time step, density, temperature and velocity are all decomposed to different components for our desired number of dimensions. We solve first for ρ and $\mathcal{T}$ based on their corresponding values and previous-time velocity field. Then we utilize our model to calculate the $\mathbf{f}_{buoy}$, given the initial value of α, β and $\mathcal{T}_{amb}$. It follows that we advect and apply this force to our velocities and obtain an intermediate field. Finally we compute the pressure term and rely on this to project our temporary $\mathbf{u}^*$ value onto the corrected $\mathbf{u}^{n+1}$ for the upcoming time step.

For each time iteration, The algorithm can be rewritten as the following:

(1) Advect density and temperature to ρ^* and $\mathcal{T}^*$
(2) Transform forward to Fourier space as $\hat{\rho}^*$ and $\hat{\mathcal{T}}^*$
(3) Diffuse using calculated matrices
(4) Transform backward to real space as ρ^{n+1} and $\mathcal{T}^{n+1}$
(5) Calculate external force $\mathbf{f}_{buoy}$
(6) Advect velocity and apply external force to it as $\mathbf{u}^*$
(7) Transform forward to Fourier space as $\hat{\mathbf{u}}^*$
(8) Diffuse using calculated matrices
(9) Compute the pressure and subtract its gradient from the intermediate velocity to obtain $\hat{\mathbf{u}}_2^*$
(10) Transform backward to real space as $\mathbf{u}^{n+1}$

4 RESULTS

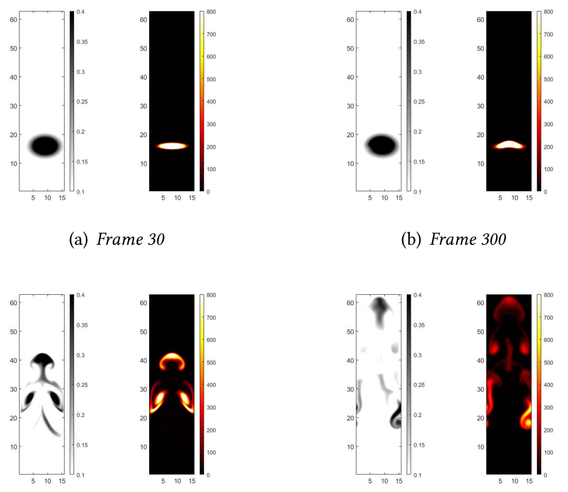
(a) *Frame 30* (b) *Frame 300* (c) *Frame 3500* (d) *Frame 7000*

Fig. 2. Cross section of 3D smoke simulated using our method, where left part represents density and right part for temperature. The data is generated in C and the pictures are drawn in Matlab. $\alpha = 0.01$, $\beta = 0.001$, and $\mathcal{T}_{amb} = 20.0$. Resolution is $64 \times 256 \times 64$ pixels. All other parameters are set to be the same.

We simulate the smoke plume with a spherical source of fluid subject to a density-temperature related buoyancy on y-axis, producing the phenomena of turbulent breakup of the rising smoke column, see Fig.2 for its cross section at different time step. We record the computational time for each simulation in the table below. The code is running uniquely with GCC.

Resolution	Num Time-Steps	Δt	Tot Comp.Time
$64 \times 256 \times 64$	30	0.005	~11 sec
$64 \times 256 \times 64$	300	0.005	~110 sec
$64 \times 256 \times 64$	3500	0.005	~1280 sec
$64 \times 256 \times 64$	7000	0.005	~2500 sec

The time we recorded includes all costs such as memory allocation, initial condition setup and file output. It is worth noting that our method has a decent time of computation in C. Compared to running in Matlab, it takes only one-third of the time to complete the simulation, whereas the other costs for instance approximately 310 seconds to generate 300 frames with same parameters, and accounted only for iterations.

5 CONCLUSIONS

We follow the conventional advection-projection scheme but apply Fourier spectral method to solve the Navier-Stokes equations. This gives an extremely clear algorithmic structure and provides with significant potential to stylize and generalize. Meanwhile, the computational cost is satisfactory as we are able to generate scenes with suitable resolution with relatively great efficiency.

When attention is paid to emulate delicate and details-preserved real-world fluids, the upcoming work is computer graphics treatment to our raw data, including but not limited to ray-tracing and rendering. The ultimate goal is to produce high-quality fluid flow animations.

Moreover, we are always willing to promote our method to the next-level. The Fourier spectral method provides the foundation for some novel real-time simulation techniques in computer gaming because of its decent computational costs, and on the other hand it is able to supplement to some computer graphics methods in pursuit of infinite resolution. We believe our method is worth further exploring.

REFERENCES

R. Fedkiw, J. Stam, and H. W. Jensen. 2001. Visual Simulation of Smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 15–22. https://doi.org/10.1145/383259.383260