# CSE291 HW 4: Fluids

Ritoban Roy-Chowdhury

June 12, 2023

## Governing Equations

The project simulates a fluid in 2 dimensions, using a vortex particle method. In general, the evolution of a fluid is given by the Navier-Stokes equation,

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \nabla_{\mathbf{u}}\mathbf{u}\right) = -\operatorname{grad} p + \mu\Delta u. \tag{1}$$

where $\mathbf{u}$ is the velocity field of the fluid, $\rho$ is its density, $p$ is the pressure, and $\mu$ is the viscosity coefficient. In the case of *incompressible* fluid simulation, we also have the requirement that $\operatorname{div}\mathbf{u} = 0$. If we ignore the viscosity term (so we're simulating an inviscid fluid), we get the incompressible Euler equations, which is what I simulate in this project:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \nabla_{\mathbf{u}}\mathbf{u}\right) = -\operatorname{grad} p \qquad \operatorname{div}\mathbf{u} = 0. \tag{2}$$

The key idea of the vortex particle method is to instead consider the *vorticity*, which is generally defined as $\omega = \operatorname{curl}\mathbf{u}$, but is better understood as the voriticity 2-form $\omega = \mathbf{d}(\mathbf{u}^\flat)$. Then, the incompressible Euler equations are equivalent to the vorticity equation

$$\frac{\partial \omega}{\partial t} + \mathcal{L}_{\mathbf{u}}\omega = 0. \tag{3}$$

## Vortex Particle Method

The idea of the vortex particle method is to store $n$ particles, with positions $\phi_p$, each of which represents a point vortex. The positions of these point vortices are updated using the Biot-Savart law, using the differential equation

$$\frac{d}{dt}\phi_p = \sum_{q=1,q\neq p}^{n} \frac{\kappa_q}{2\pi}\frac{R^{90^\circ}(\phi_p - \phi_q)}{|\phi_p - \phi_q|^2}. \tag{4}$$

Let $\mathbf{v}_p(\mathbf{x})$ denote the function on the right-hand-side of that equation, for the location $\phi_p = \mathbf{x}$. I discretize this differential equation using RK4. In particular, the position at time $n+1$ is

computed from the position at time $n$ as

$$k_1 = \mathbf{v}_p(\phi_p) \tag{5}$$

$$k_2 = \mathbf{v}_p\left(\phi_p + \frac{\Delta t}{2}k_1\right) \tag{6}$$

$$k_3 = \mathbf{v}_p\left(\phi_p + \frac{\Delta t}{2}k_2\right) \tag{7}$$

$$k_4 = \mathbf{v}_p(\phi_p + \Delta t k_3) \tag{8}$$

$$\phi_p^{(n+1)} = \phi_p^n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{9}$$

$$\tag{10}$$

This allows us to simulate several vortex particles moving around according to the Biot-Savart law.

In order to get a nice visual of a fluid, I also simulate a colored "dye" being advected by the fluid velocity, using semi-Lagrangian advection. Specifically, the dye is represented as a $800 \times 800$ grid of pixels, where each pixel stores a color. At each timestep, we compute the world-space position of the pixel, and then perform an RK4 backtrace using the above algorithm (e.g. using $-\Delta t$ instead of $\Delta t$), except without skipping any particles in the summation for computing the velocity (e.g. withou the $q \neq p$ condition in the summation above). Then, once we have the backtraced position, we use bilinear interpolation from the neighboring four pixels to compute a color, and set the color of the present pixel to the interpolated color.

This simulates a dye being passively advected by the Biot-Savart velocity field generated by the vortex particles.

## Example

In the particular demo presented here, I begin the simulation with 20 vortex particles, randomly initialized to locations in $[0, 1]^2$, with strengths $\kappa_p$ also randomy initialized in $[0, 1]$. The vortex particles are then time-stepped forward with $\Delta t = 0.01$.

The dye is initialized in the LCH color space with the following values (which I just decided looked nice by experimentation)

$$H = 360 \sin(3x) \cos(5y) \tag{11}$$

$$C = \frac{100}{2}(|\cos(7x)||\cos(4y)| + 1) \tag{12}$$

$$L = \frac{100}{2}(|\cos(2x)||\cos(3y)| + 1) \tag{13}$$

It is then stepped forward as described above using semi-Lagrangian advection, using an RK4 backtrace to find the previous position of the dye.

## Implementation Details

The dye colors are stored in the LCH color space, because performing color-interpolation in the RGB color space causes the colors to blend together and slowly tend towards a brown-grey. Instead, performing the interpolation in the LCH color space generally preserves the percieved lightness

and saturation of the colors. Interestingly, the conversion from LCH to RGB is actually quite a significant part of the runtime of the program.

When performing semi-Lagrangian advection for the dye, if the advection requires interpolating data that is off of the grid where we have dye values, then I simply take the nearest grid value. This unfortunately sometimes causes large swaths of a uniform color to come into the simulation, but at least it still gives generally decent looking results. A better approach might be to do the whole simulation on a periodic domain, but that might cause other visual artifacts.

I also very simply multithreaded the advection of the dye using the Rayon library – this lets the simulation run *much* faster, since the runtime is dominated by the advection of the dye anyway (as opposed to moving the vortex particles around).