Course Note: Grid Based Fluid Simulation

USC CS599 Physically Based Modeling for Interactive Simulation and Games
Instructor: Prof. Jernej Barbic
Date: Mar. 11 2010
Collator: Yili Zhao
(This record is based upon my original notes, my understanding on this problem and some additional materials. Any comments, suggestions, and of course, the inevitable critique are highly welcomed.)

Overview of numerical Simulation

The simulation of fluid is typically based on solving the *Navier-Stokes* (NS) equations, which describe the fluid in terms of a continuous velocity field $\vec{u}$ and a pressure p [1]. They are usually written as:

$$\frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u} + \nu \nabla \cdot \nabla \vec{u} - \frac{1}{\rho} \nabla p + \vec{f} \qquad (1)$$

$$\nabla \cdot \vec{u} = 0 \qquad (2)$$

Equation (1) is a vector equation called "**momentum equation**" which ensures the momentum of the fluid is preserved. Though it seems a little complicated, we can split it up into individual parts and solve each one separately, which significantly reduced the problem's difficulty. Generally, this formula could be divided into 4 parts, which are advection part, body force part, viscosity part and pressure part. The viscosity part and the body force part are very easy to handle. How to compute the advection has been discussed in previous class. The last and most complex term is the pressure part.

In order to solve it, we put our focus on Equation (2) named "**incompressibility condition**". Pick an arbitrary chunk of fluid at some instant in time. Denote this volume Ω and its boundary surface $\partial\Omega$. (See Figure 1)

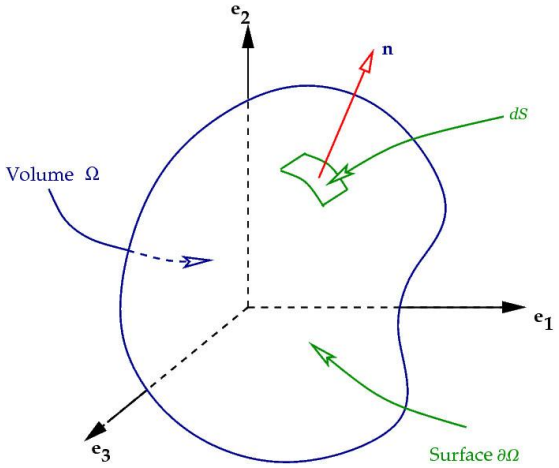


Figure 1: Volume for application of the divergence theorem [4]

We could measure how fast the volume of this chunk is changing by integrating the normal component of its velocity around the surface area, that is,

$$\frac{dV}{dt} = \iint_{\partial\Omega} \vec{u} \cdot \vec{n}\, ds \qquad (3)$$

For **incompressible** fluid, this rate of change should be **zero**.

By using Gauss Theorem, the equation (3) could be changed to a volume integral,

$$\frac{dV}{dt} = \iint_{\partial\Omega} \vec{u} \cdot \vec{n}\, ds = \iiint_{\Omega} \quad \nabla \cdot \vec{u} = 0 \quad (4)$$

This equation should be true for any choice of Ω, thus we obtain,

$$\nabla \cdot \vec{u} = 0$$

This is equation (2), the incompressibility condition.

A vector field that satisfies the equation (2) is "**divergence-free**" [2]. The advection should only be done in such kind of field. However given an arbitrary vector field $\vec{w}$, the equation (2) may not be guaranteed, i.e. $\nabla \cdot \vec{w} \neq 0$. We have to implement *Helmholtz-Hodge Decomposition* on $\vec{w}$ [3], which leads to:

$$\vec{w} = \vec{u} + \nabla q \qquad (5)$$

In equation (5) $\vec{u}$ is divergence-free and q is scalar field. (*Note: instead of q, the professor used p in equation (5) during the class. However as a general equation here, in order not to be confused with pressure p in equation (1), I changed the notation.)*

By multiplying both sides of equation (5) by gradient operator,

$$\nabla \cdot \vec{w} = \nabla \cdot \nabla q \qquad (6)$$

This is a Poisson equation for the scalar filed q with certain boundary conditions.

Things get clear now that given the velocity field computed from the previous steps we compute the pressure values in a way that ensures equation (2) holds. It turns out that a Poisson equation with certain boundary conditions has to be solved for the pressure [1]. Once a correct pressure is obtained, the velocity field is adjusted to maintain the divergence free condition and describes the state of the fluid for the next time step [1].

Discretization

Our environment contains an arbitrary distribution of fluid, and submerged or semi-submerged obstacles. The computation domain is first divided into a fixed rectangular grid aligned with a Cartesian coordinate system. Without lose generality, we illustrate the method in 2D scene. (See Figure 2)

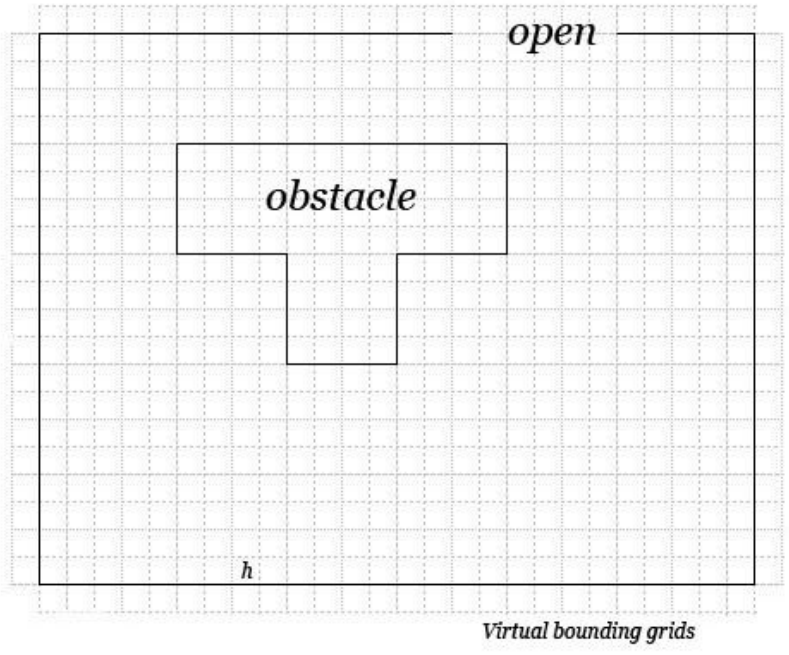


Figure 2: Discretized computation domain

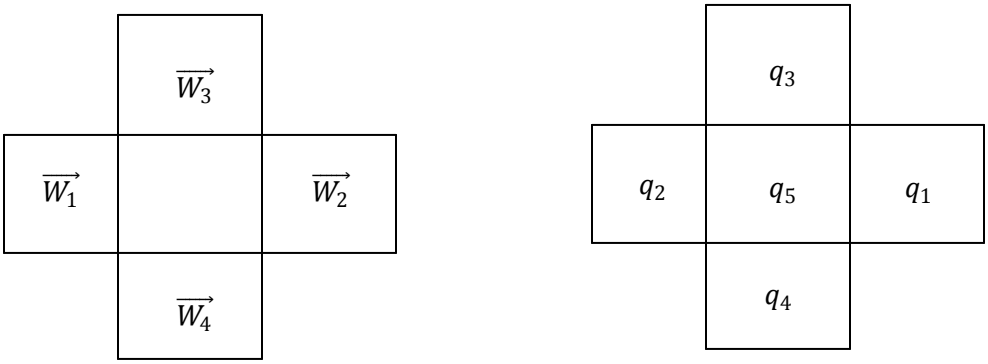


Figure 3: 4-connected grids

Suppose the length of each unit grid is *h*,

$$\nabla \cdot \vec{w} = \frac{\vec{w}_{1,x} - \vec{w}_{2,x}}{2h} + \frac{\vec{w}_{3,y} - \vec{w}_{4,y}}{2h} \qquad (7)$$

$$\nabla \cdot \nabla q = \frac{p_1 - 2p_5 + p_2}{h^2} + \frac{p_3 - 2p_5 + p_4}{h^2} \qquad (8)$$

We could set up the linear equation based on equation (6) (7) (8) and solve it using conjugate gradient method.

Once q is solved,

$$\vec{u} = \vec{w} - \nabla q \qquad (9)$$

Boundary conditions

Boundary conditions need only be checked once at the beginning of iteration.

A boundary is an interface between the fluid and a solid obstacle, or between the fluid and atmosphere, or a point at which fluid flows into or out of the system. Solid obstacles and the atmosphere are treated as fluid, but with special properties that remain constrained throughout the calculation.

The component of fluid velocity normal to the face of obstacle is 0, e.g. $u_x = 0$. In the case of a **non-slip** obstacle, the tangential velocity at the boundary is also 0. While in a **free-slip** case, the tangential velocity keeps unchanged at the boundary.

As we are always considering 4-connected area during the calculation, we add virtual bounding grids around the computation domain for convenience and set their values accordingly.

For the purpose of tracking fluid position and setting up the boundary conditions, the finite difference grids are labeled as [5], (See Figure 4 for example)

- *Empty E*, a cell containing no particles.
- *Surface S*, a cell containing at one particle that is adjacent to an empty cell.
- *Full F*, a cell containing at least one particle that is not a surface cell.

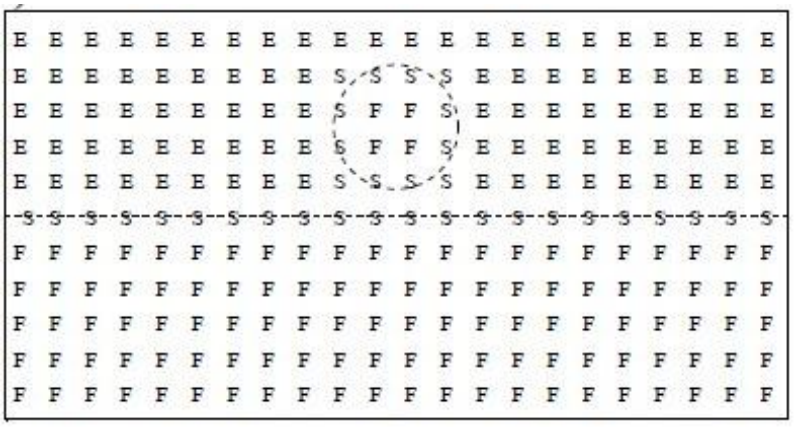
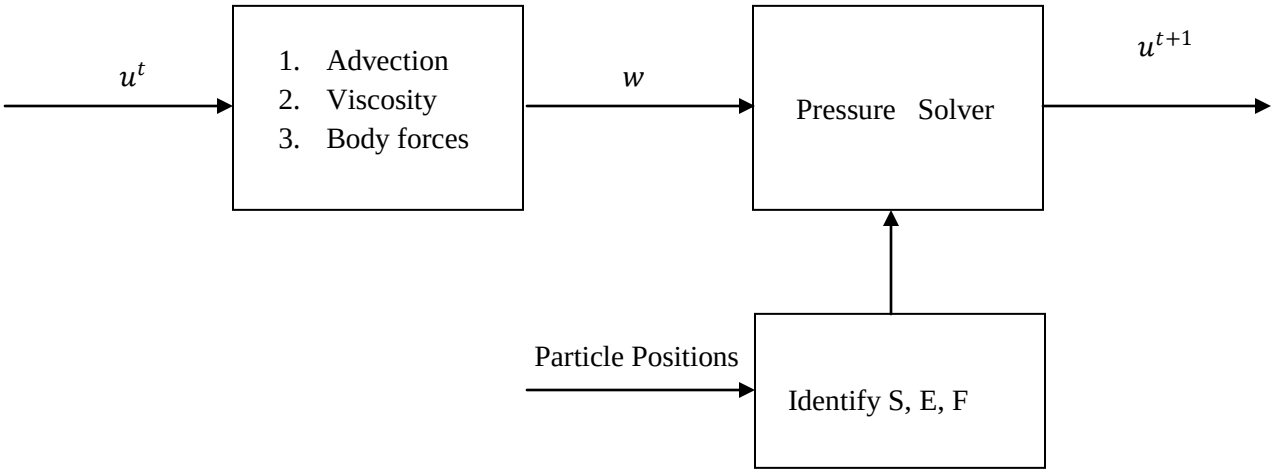


Figure 4: labeled computation domain

Please check [5] for more information on configuration of cells.

In conclusion, the iteration process during simulation could be summarized as follows:



References:

[1] Muller M., Stam J., James D., Thurey N.: Real time physics: class notes. In SIGGRAPH ’08: ACM SIGGRAPH2008 Course, pp. 1–90.

[2] MÜLLER M., R. Bridson. Fluid Simulation, In SIGGRAPH’07: ACM SIGGRAPH2007 Course

[3] Stam J.: Stable fluids. In Proc. of ACM SIGGRAPH1999, pp. 121–128.

[4] http://en.wikiversity.org/wiki/Introduction_to_Elasticity/Vectors

[5] N. Foster, D. Metaxas. Realistic Animation of Liquids. Graphical Models and Image Processing, 58(5):471–483, 1996.