Incompressible Fluid Simulation: A Comparison

YUMENG HE, University of Southern California, USA
HSIN LI, University of Southern California, USA
YUCHEN CHEN, University of Southern California, USA
XU CHEN, University of Southern California, USA

May 8, 2025

Abstract :

Our project is a 2D incompressible fluid simulation implemented in C++ with visualization using OpenGL and GLUT. The main objective is to compare the performance, visual behavior, and numerical characteristics of different fluid simulation methods, including: Grid-based (Stable Fluids), Particle-based (SPH), Particle-In-Cell (PIC),hybrid PIC/FLIP method(PIC/FLIP), Affine Particle-In-Cell (APIC) This simulation provides a visual and algorithmic comparison of each method's strengths and weaknesses.

Index Terms: fluid simulation, incompressible, particle, grid, hybird

1 Introduction

Fluid simulation is a central topic in physics-based graphics and engineering. Researchers study two broad classes of flow. Compressible fluids—such as smoke, fire, or drifting snow—change density as they move. Incompressible fluids—such as water—preserve volume. Our project narrows its focus to incompressible flow because it underpins many game and film effects.

Scientists have pursued fluid solvers for more than three decades. Early work in the 1990 s split along two lines. Grid-based methods stored velocity on fixed cells and solved pressure on a lattice. Particle methods—notably Smoothed Particle Hydrodynamics (SPH)—tracked discrete parcels of mass. Each line had limits: grids diffused small details, while pure particles struggled with volume loss and boundary handling.

Around 2000, hybrid techniques emerged. Particle-In-Cell (PIC) used a grid for forces and particles for advection. FLIP kept the same layout but reduced numerical damping. Material Point Method (MPM) added elastoplastic behavior for snow-like media. Affine Particle-In-Cell (APIC) later improved rotational fidelity by carrying local affine velocity. These methods mix Eulerian and Lagrangian views to balance stability and detail.

Our project builds an interactive framework that implements five representatives: Stable Fluids (grid), SPH (particle), PIC, hybrid PIC/FLIP, and APIC. We run every solver on the same domain, time step, and boundary conditions. We then measure speed, memory use, and visual artifacts. The side-by-side view reveals each method's trade-off between diffusion, noise, and stability, and helps artists choose the right tool for a desired effect.

1.1 Contribution

This project as the follow contributions.

- The codebase supports five fluid solvers behind one interface. Users can swap methods with a single flag.
- The viewer renders density, velocity, and vorticity in real time. It uses GLUT for portability.
- We fix domain size, time step, and boundary conditions across all tests. This isolates algorithmic differences.
- We capture signature phenomena such as diffusion, particle clumping, and energy drift. Screenshots and videos illustrate each effect.

2 Background

Fluid simulation typically relies on solving the Navier-Stokes equations, which describe fluid motion as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{1}$$

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

where $\mathbf{u}$ is the velocity field, p is the pressure, ρ is the density, ν is the kinematic viscosity, and $\mathbf{f}$ represents external forces like gravity or user input. The second equation enforces incompressibility.

2.1 Grid-based (Stable Fluids)

Grid-based methods store velocity and pressure fields on a fixed Eulerian grid. The Stable Fluids method proposed by Stam **Stam 1999** employs an implicit numerical scheme that guarantees stability at the cost of numerical diffusion. This approach involves four primary steps: advection, diffusion, force application, and pressure projection to ensure incompressibility. Although easy to implement and stable, this method diffuses small-scale features rapidly, causing loss of detail.

2.2 Particle-based (SPH)

Smoothed Particle Hydrodynamics (SPH) is a purely Lagrangian, particle-based technique. It represents fluid with discrete particles that carry fluid properties such as density and velocity **Monaghan1992SPH**. Particle interactions are computed using smoothing kernels, enabling flexible boundary handling and adaptive resolution. However, SPH often struggles with preserving volume and can produce noisy visual artifacts, especially with low particle counts.

2.3 Hybrid Methods

Hybrid approaches blend Eulerian grids and Lagrangian particles, seeking a balance between stability, accuracy, and visual realism. Notable hybrid methods include:

Particle-In-Cell (PIC): PIC **Harlow1964PIC** transfers velocities from particles to a grid to compute pressure and forces, then advects particles using the grid velocities. It offers stability but introduces significant numerical damping.

FLuid Implicit Particle (FLIP): An improvement over PIC, FLIP **Brackbill1986FLIP** reduces numerical damping by transferring velocity changes, rather than absolute velocities, from grid to particles.

Affine Particle-In-Cell (APIC): APIC **Jiang2015APIC** further improves rotational and detailed motion preservation by storing affine velocity transformations for each particle, mitigating excessive dissipation seen in PIC/FLIP methods.

Material Point Method (MPM): Extending PIC, MPM **Sulsky1995MPM** simulates elastoplastic and granular materials by integrating material deformation through particle-grid interactions.

Other advanced hybrid variations include:

- **PolyPIC Fu2017PolyPIC**, which uses polynomial velocity reconstruction to reduce numerical dissipation.
- **MLS-MPM** (Moving Least Squares MPM) **Hu2018MLSMPM**, enhancing accuracy by employing MLS interpolation.
- **Impulse PIC Jiang2021ImpulsePIC**, improving collision handling by explicitly resolving impulses at boundaries.

These hybrid methods significantly advance fluid simulation, enabling realistic visualization with reduced artifacts and increased computational stability.

3 Methods

To accomplish our project goals, we implemented five distinct 2D incompressible fluid simulation methods—Stable Fluids, SPH, PIC, PIC/FLIP, and APIC—using C++ for the core simulation and OpenGL with GLUT for real-time visualization. Each method was developed independently based on its underlying physical principles and algorithmic structure. We focused on observing and comparing the visual behavior and numerical characteristics of each simulation through qualitative analysis. The following sections describe the implementation details and key observations for each method.

Tools and Learning We used C++ for simulation logic and OpenGL with GLUT for real-time visualization across all simulation methods. The Eigen library was employed for efficient matrix operations, particularly for APIC and FLIP methods where affine velocity matrices were involved.

Throughout the project, we learned how to structure particle-grid transfer systems, implement spatial neighborhood queries using a sorting grid, and visualize thousands of particles in real time. We also gained practical experience with parallel programming, numerical debugging, and enforcing boundary conditions on staggered MAC grids.

Course Content Reference We applied key concepts from the course, including hybrid fluid simulation methods (PIC, FLIP, APIC), particle-grid transfers, SPH kernel functions, external forces, and pressure projection. These topics directly guided our simulation and implementation strategy.

3.1 Particle

We used a particle fluid simulation method developed by Monaghan **Monaghan1992SPH**, called Smoothed Particle Hydrodynamics (SPH). Our code focused on modeling using SPH formulations with fluid forces such as pressure and viscosity. SPH is an interpolation method that evaluates field quantities of each particle based on its local neighborhood using radial symmetrical smoothing kernels.

3.1.1 Algorithm

The core steps of the SPH particle fluid simulation is summarized in Algorithm 1.

Algorithm 1 SPH Particle Update Loop

```
1:  // Compute density and pressure
2:  for each particle $i$ do
3:      for each neighboring particle $j$ do
4:          Compute distance
5:          if Within kernel radius then
6:              Add density contribution $\rho_i = m_j \cdot W(r_{ij}, h)$
7:          end if
8:      end for
9:      Compute pressure from density $P = k_p(\rho - \rho_0)$
10: end for
11: // Compute forces on each particle
12: for each particle $i$ do
13:     Initialize $f_p \leftarrow 0, f_v \leftarrow 0$
14:     for each neighboring particle $j$ do
15:         Compute distance
16:         if Within kernel radius then
17:             Compute pressure force contribution
18:             $f_p = m_j \cdot \frac{p_i + p_j}{2\rho_j} \cdot \nabla W(r_i j, h)$
19:             Compute viscosity force contribution
20:             $f_p = m_j \cdot \frac{v_j - v_i}{2\rho_j} \cdot \nabla^2 W(r_i j, h)$
21:         end if
22:     end for
23:     Compute gravity force contribution $f_g = G \cdot \frac{m_i}{\rho_i}$
24:     Total force on particle $f_i = f_p + f_v + f_g$
25: end for
26: // Integrate velocity and update positions
27: for each particle $i$ do
28:     Update velocity $v_i^{t+\Delta t} = v_i^t = \Delta t \cdot \frac{f_i^t}{\rho_i}$
29:     Update position $x_i^{t+\Delta t} = x_i^t + \Delta t \cdot v_i^{t+\Delta t}$
30:     if position $x_i$ hits domain boundary then
31:         Dampen velocity
32:         Clamp position to boundary
33:     end if
34: end for
```

3.1.2 Intermediate Results and Diagrams

We initialized our SPH simulation with a slight jitter to each particle's initial positions. This is because SPH evalautes its fields based on its neighbors, and if particles left and right are evenly spaced, the horizontal forces will cancel out perfectly. This would lead to the particles only bouncing vertically.

3.2 APIC

To implement the APIC method, we aimed to simulate incompressible fluid behavior with both stability and visual richness. Compared to PIC or FLIP, APIC introduces an affine velocity field per particle to better capture rotational and shear motion, which helps reduce excessive numerical dissipation and jittering effects.

3.2.1 Algorithm The core steps of the APIC method are summarized in Algorithm 2. This method extends the standard PIC approach by introducing an affine velocity matrix for each particle, which allows capturing local rotational and shear motions more accurately.

Algorithm 2 APIC Particle Update Loop

```
1: for each particle $p$ do                                  ▷ Particle to Grid (P2G)
2:     for each neighboring grid node $g$ do
3:         Compute weight $w_{pg}$ and offset $\mathbf{d} = x_g - x_p$
4:         Transfer velocity: $v_g \leftarrow v_g + w_{pg} \cdot (v_p + C_p \cdot \mathbf{d})$
5:         Transfer mass: $m_g \leftarrow m_g + w_{pg}$
6:     end for
7: end for
8: for each grid node $g$ do                                 ▷ Grid Operations(Add Forces)
9:     if $m_g > 0$ then
10:        Normalize: $v_g \leftarrow \frac{v_g}{m_g}$
11:    end if
12:    Apply gravity: $v_g \leftarrow v_g + \Delta t \cdot g$
13:    Enforce boundary conditions on $v_g$
14: end for
15: for each particle $p$ do                                 ▷ Grid to Particle (G2P)
16:    Initialize: $v_p \leftarrow 0, C_p \leftarrow 0$
17:    for each neighboring grid node $g$ do
18:        Compute weight $w_{pg}$ and offset $\mathbf{d} = x_g - x_p$
19:        Interpolate velocity: $v_p \leftarrow v_p + w_{pg} \cdot v_g$
20:        Update affine matrix: $C_p \leftarrow C_p + w_{pg} \cdot v_g \otimes \mathbf{d}$
21:    end for
22:    Update position: $x_p \leftarrow x_p + \Delta t \cdot v_p$
23: end for
```

3.2.2 Intermediate Results and Diagrams Figure 1 shows the particle distribution in our APIC simulation using 4000 particles. The APIC method retains coherent motion and prevents clumping or artificial viscosity, which is often observed in simpler schemes like PIC. The particles settle smoothly while preserving rotational features due to the affine velocity transfer.

4 Conclusion

Through this project, we gained hands-on experience implementing a variety of fluid simulation methods, including particle-based (SPH), grid-based (Stable Fluids), and hybrid approaches (PIC, FLIP, and APIC). We deepened our understanding of pressure projection, velocity interpolation, particle-grid transfers, and fluid behavior visualization. On the implementation side, we learned to work with OpenGL and GLUT for real-time rendering, and used the Eigen library for efficient linear algebra operations. We also practiced debugging and tuning numerical simulations, and managing complexity within a modular C++ codebase.

Team Contributions Xu Chen was responsible for the OpenGL-based visualization system and implemented the APIC method. Yumeng He contributed to both the particle system and grid-based simulation components. Irene Li worked on particle and grid simulations. Yuchen Chen implemented the PIC and FLIP methods and also contributed to grid development.

Future Work This project has sparked our interest in computer graphics and physically based animation. In the future, we hope to explore more advanced topics such as 3D fluid simulation, GPU acceleration, and real-time rendering techniques.

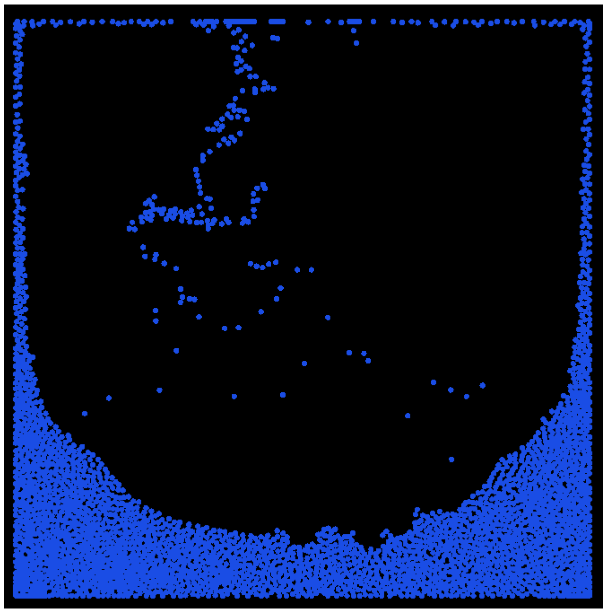

Figure 1. Particle distribution in APIC simulation with 4000 particles. The affine velocity transfer effectively preserves fine rotational structures and avoids excessive dissipation compared to PIC.

Acknowledgements

This project received support during the CSCI580 course, instructed by Professor XXX at the University of Southern California, USA.