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.

Acknowledgements

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