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 9, 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, 2023) 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 (Monaghan, 1992). 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 (Tskhakaya et al., 2007) 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 (Brackbill et al., 1988) reduces numerical damping by trans-

ferring velocity changes, rather than absolute velocities, from grid to particles.

Affine Particle-In-Cell (APIC): APIC (Jiang et al., 2015) 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 (Bardenhagen et al., 2000) simulates elastoplastic and granular materials by integrating material deformation through particle-grid interactions.

Other advanced hybrid variations include:

- **PolyPIC** (Fu et al., 2017), which uses polynomial velocity reconstruction to reduce numerical dissipation.
- **MLS-MPM** (Moving Least Squares MPM) (Hu et al., 2018), enhancing accuracy by employing MLS interpolation.
- **Impulse PIC** (Feng et al., 2022), 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 Grid

The first method implemented in our project is the Stable Fluids method introduced by Stam (Stam, 2023). This grid-based Eulerian approach uses a fixed discretized grid to represent fluid properties, such as velocity and density fields. The method ensures unconditional stability at the expense of numerical diffusion, making it robust for real-time applications.

The numerical solver follows four main computational steps:

1. **Add Source**: Introduce external quantities (density, velocity) into the simulation. Each cell's value is incremented by a source term scaled by the simulation timestep:
$$x_{i,j} \leftarrow x_{i,j} + \Delta t \cdot s_{i,j}$$
2. **Diffuse**: Account for viscosity by spreading fluid properties across the grid. This step employs iterative Gauss-Seidel or Jacobi methods to solve the diffusion equation implicitly:
$$\frac{x_{i,j}^{t+1} - x_{i,j}^{t}}{\Delta t} = \nu \nabla^2 x_{i,j}^{t+1}$$
3. **Project**: Enforce incompressibility by adjusting the velocity field to be divergence-free. The divergence is computed, a pressure field is solved via iterative Jacobi relaxation, and then the pressure gradient is subtracted from the velocity:
$$\nabla^2 p = \nabla \cdot \mathbf{u}, \quad \mathbf{u} \leftarrow \mathbf{u} - \nabla p$$
4. **Advect**: Transport fluid properties through the velocity field. Each grid cell is traced backward in time along the velocity field, and bilinear interpolation reconstructs values:
$$x_{i,j}^{t+1} = x(\mathbf{p} - \Delta t \cdot \mathbf{u}(\mathbf{p}, t), t)$$

Finally, appropriate **Boundary Conditions** are applied after each step: velocity components are inverted at solid boundaries, and scalar fields maintain values by copying adjacent interior cells. These operations are encapsulated within the functions `vel_step()` and `dens_step()`, which are sequentially called in the main simulation loop (`simulation()`).

3.2 Particle

We used a particle fluid simulation method developed by Monaghan (Monaghan, 1992), 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.

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

3.2.2 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.

3.3 PIC

The Particle-In-Cell (PIC) method uses both particles and grids for fluid simulation. Each simulation step involves five primary phases:

1. **Transfer to Grid**: Particle velocities are transferred to nearby grid cells using B-spline weighting functions. The compact quadratic B-spline kernel used is:
$$w(r) = \begin{cases} 0.75 - r^2, & 0 \le r < 0.5 \\ 0.5 \cdot (1.5 - r)^2, & 0.5 \le r < 1.5 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Algorithm 1 SPH Particle Update Loop

// Compute density and pressure
for each particle i **do**
  for each neighboring particle j **do**
    Compute distance
    if Within kernel radius **then**
      Add density contribution $\rho_i = m_j \cdot W(r_{ij}, h)$
    end if
  end for
  Compute pressure from density $P = k_p(\rho - \rho_0)$
end for
// Compute forces on each particle
for each particle i **do**
  Initialize $f_p \leftarrow 0$, $f_v \leftarrow 0$
  for each neighboring particle j **do**
    Compute distance
    if Within kernel radius **then**
      Compute pressure force contribution
      $f_p = m_j \cdot \frac{p_i + p_j}{2\rho_j} \cdot \nabla W(r_i j, h)$
      Compute viscosity force contribution
      $f_p = m_j \cdot \frac{v_j - v_i}{2\rho_j} \cdot \nabla^2 W(r_i j, h)$
    end if
  end for
  Compute gravity force contribution $f_g = G \cdot \frac{m_i}{\rho_i}$
  Total force on particle $f_i = f_p + f_v + f_g$
end for
// Integrate velocity and update positions
for each particle i **do**
  Update velocity $v_i^{t+\Delta t} = v_i^t = \Delta t \cdot \frac{f_i^t}{\rho_i}$
  Update position $x_i^{t+\Delta t} = x_i^t + \Delta t \cdot v_i^{t+\Delta t}$
  if position x_i hits domain boundary **then**
    Dampen velocity
    Clamp position to boundary
  end if
end for

2. **Apply Gravity**: Gravity force is applied directly to vertical grid velocities. Grid velocity is updated using:

$$\vec{v}_{i,j}.y \mathrel{-}= g \cdot \Delta t \tag{4}$$

3. **Solve Pressure**: The pressure Poisson equation is solved using Jacobi iteration. Initially, divergence is calculated for each grid cell.

$$\nabla \cdot \vec{v}_{i,j} = \frac{v_{i+1,j}.x - v_{i-1,j}.x}{2} + \frac{v_{i,j+1}.y - v_{i,j-1}.y}{2} \tag{5}$$

Then, pressure values are iteratively adjusted to minimize divergence, enforcing incompressibility.

$$p_{i,j}^{(k+1)} = \frac{1}{N}\left(\sum_{\text{fluid neighbors}} p^{(k)} - \nabla \cdot \vec{v}_{i,j}\right) \tag{6}$$

where N is the number of neighboring fluid cells.

The velocity field is updated by subtracting the pressure gradient.

$$\vec{v}_{i,j}.x \mathrel{-}= \frac{p_{i+1,j} - p_{i-1,j}}{2}, \quad \vec{v}_{i,j}.y \mathrel{-}= \frac{p_{i,j+1} - p_{i,j-1}}{2} \tag{7}$$

4. **Transfer Back to Particles**: Updated grid velocities are interpolated back onto particles.

$$\vec{v}_p = \sum_{(i,j)} w_{(i,j)\to p} \cdot \vec{v}_{i,j} \tag{8}$$

5. **Move Particles**: Particles are advected according to their updated velocities. Boundary conditions are enforced by repositioning particles inside the domain and setting boundary normal velocities to zero.

$$\vec{x}_p \mathrel{+}= \vec{v}_p \cdot \Delta t \tag{9}$$

3.4 PIC/FLIP Implementation

The PIC/FLIP hybrid method follows a similar pipeline but differs in how particle velocities are updated:

1. Before applying gravity, the current grid velocity is stored.
2. After solving the pressure, the difference between the new and old grid velocities is computed.
3. Particle velocities are updated using a blend of PIC and FLIP:

$$\begin{aligned}&\text{blended\_velocity}\\ &= \text{particle\_velocity}\\ &+ \text{flip\_ratio} \cdot (\text{new\_grid\_velocity} - \text{old\_grid\_velocity})\end{aligned} \tag{10}$$

where a flip_ratio of 0 corresponds to pure PIC, and values approaching 1 resemble FLIP.
4. Particles are then advected in the same manner as the PIC method, including boundary handling.

3.5 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.

3.5.1 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.

3.5.2 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 Results

4.1 APIC

We evaluated the performance of the APIC method with different particle counts. Figure 2 shows simulation snapshots at three resolutions—1000, 4000, and 8000 particles—demonstrating how the

Algorithm 2 APIC Particle Update Loop

for each particle p **do** ▷ Particle to Grid (P2G)
  for each neighboring grid node g **do**
    Compute weight w_{pg} and offset $\mathbf{d} = x_g - x_p$
    Transfer velocity: $v_g \leftarrow v_g + w_{pg} \cdot (v_p + C_p \cdot \mathbf{d})$
    Transfer mass: $m_g \leftarrow m_g + w_{pg}$
  end for
end for
for each grid node g **do** ▷ Grid Operations(Add Forces)
  if $m_g > 0$ **then**
    Normalize: $v_g \leftarrow \frac{v_g}{m_g}$
  end if
  Apply gravity: $v_g \leftarrow v_g + \Delta t \cdot g$
  Enforce boundary conditions on v_g
end for
for each particle p **do** ▷ Grid to Particle (G2P)
  Initialize: $v_p \leftarrow 0$, $C_p \leftarrow 0$
  for each neighboring grid node g **do**
    Compute weight w_{pg} and offset $\mathbf{d} = x_g - x_p$
    Interpolate velocity: $v_p \leftarrow v_p + w_{pg} \cdot v_g$
    Update affine matrix: $C_p \leftarrow C_p + w_{pg} \cdot v_g \otimes \mathbf{d}$
  end for
  Update position: $x_p \leftarrow x_p + \Delta t \cdot v_p$
end for

method handles fluid detail, stability, and distribution over time. Each subfigure compares the final state of the fluid, and highlights how increasing the number of particles leads to smoother, more detailed results.

5 Discussion

Table 1
Comparison of Fluid Simulation Methods

Method	Pro	Con
Grid	Fast (real time) Unconditionally stable	Loss of detail Not physically accurate
Particle	Fast	Limitation of input particle position Particle collapse
PIC	Stable	High numerical dissipation Particles lose energy quickly
PIC/FLIP	More realistic and dynamic motion	Less stable Needs tuning
APIC	Preserves rotation	More complex Slower

5.1 Method Contributions

5.1.1 APIC The APIC method contributed most significantly to the visual quality of our simulations. By introducing an affine velocity field per particle, APIC preserves both rotational motion and local deformation, resulting in smoother, more detailed fluid behavior. Compared to PIC and FLIP, it produced the most visually stable and coherent results, especially at higher particle counts. The use of affine velocity transfer also reduced numerical dissipation and prevented particle clumping, leading to more realistic motion.

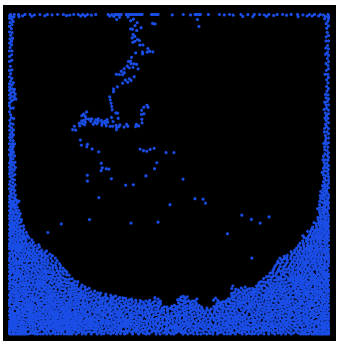


Figure 1. Particle distribution in APIC simulation with 4000 particles.

5.2 Strengths

5.2.1 APIC APIC preserved rotational motion and fine detail better than other methods. It produced smooth and stable results even with 10000 particles. Compared to PIC and FLIP, it avoided both dissipation and noise, leading to visually realistic fluid behavior.

5.3 Limitations

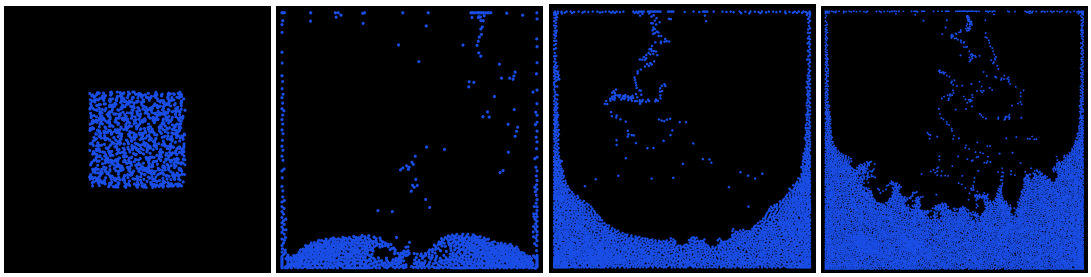
5.3.1 APIC APIC is computationally expensive due to affine matrix operations and additional interpolation. Performance drops at high particle counts, and the method becomes less suitable for real-time applications. It is also more complex to implement than PIC or FLIP, requiring careful handling of matrix math, boundary conditions, and velocity transfers to avoid instability.

6 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,



(a) 1000 initial particles **(b)** 1000 particles **(c)** 4000 particles **(d)** 8000 particles

Figure 2. Comparison of APIC simulation results with increasing particle counts. (a) shows the initial particle configuration, where all particles are placed in the center of the domain. (b)–(d) show the simulation at the moment particles start to fall under gravity.

GPU acceleration, and real-time rendering techniques.

Acknowledgements

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

References

Bardenhagen, SG, JU Brackbill, and Deborah Sulsky (2000). "The material-point method for granular materials". In: *Computer methods in applied mechanics and engineering* 187.3-4, pp. 529–541.

Brackbill, Jeremiah U, Douglas B Kothe, and Hans M Ruppel (1988). "FLIP: a low-dissipation, particle-in-cell method for fluid flow". In: *Computer Physics Communications* 48.1, pp. 25–38.

Feng, Fan et al. (2022). "Impulse fluid simulation". In: *IEEE Transactions on Visualization and Computer Graphics* 29.6, pp. 3081–3092.

Fu, Chuyuan et al. (2017). "A polynomial particle-in-cell method". In: *ACM Transactions on Graphics (TOG)* 36.6, pp. 1–12.

Hu, Yuanming et al. (2018). "A moving least squares material point method with displacement discontinuity and two-way rigid body coupling". In: *ACM Transactions on Graphics (TOG)* 37.4, pp. 1–14.

Jiang, Chenfanfu et al. (2015). "The affine particle-in-cell method". In: *ACM Transactions on Graphics (TOG)* 34.4, pp. 1–10.

Monaghan, Joe J (1992). "Smoothed particle hydrodynamics". In: *In: Annual review of astronomy and astrophysics. Vol. 30 (A93-25826 09-90), p. 543-574.* 30, pp. 543–574.

Stam, Jos (2023). "Stable fluids". In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pp. 779–786.

Tskhakaya, David et al. (2007). "The particle-in-cell method". In: *Contributions to Plasma Physics* 47.8-9, pp. 563–594.