2D FLIP/PIC fluids:

A hybrid approach to fluids

JOEY MARTEN, McGill University

This paper describes the implementation of a hybrid fluid simulator using both particles and a grid in a 2D environment. This method is based on Zhu and Bridson's approach to animating sand and is referred to as the FLIP/PIC methodZhu and Bridson [2005]. The system works by treating the fluid through the Lagrangian point of view with some of the features from the Eulerian model. The Navier-Stokes equations govern most of the mechanics behind the implementation. The resulting grid velocities can be used to calculate particle velocities by interpolating between the two methods. The dynamics of the PIC method appear much more viscous than the FLIP method. By combining both of them, the end result resembles liquid water.

Code: https://github.com/FrankfurtOceanic/PICFLIP
Video: https://youtu.be/5IXsQ7yA04g

CCS Concepts: • **Computing methodologies** → Physical simulation;

Additional Key Words and Phrases: fluid, numerical integration, Eulerian, Lagrangian, Gauss Seidel, water

ACM Reference Format:
Joey Marten. 2023. 2D FLIP/PIC fluids:: A hybrid approach to fluids. 1, 1 (April 2023), 4 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Many real-world phenomena are best described as fluids. As such, there are many reasons that fluid simulations are important models to study. There are several bodies of work within engineering that use such models to predict the interaction of an object with fluids. These techniques focus on the physical and numerical accuracy of the models. In contrast, the field of computer animation favors visual fidelity to drive the film and video game industry.

Traditionally, there have been two main ways of thinking about fluids: the Lagrangian approach and the Eulerian approach. Eulerian fluids discretize the space into a grid of cells. In two dimensions, the cells are represented as squares on a lattice. 3D methods opt for voxels. Each cell contains information about the fluid such as velocity, density, temperature, etc. On the other hand, the Lagrangian procedure uses individual particles to emulate the molecules of the fluid. The process dictates how each particle moves and interacts with the other particles. The Lagrangian approach is better suited to modeling small-scale interactions. The Eulerian model would need a very high resolution to resemble the perceptible movement of small particles. However, it becomes difficult to calculate the interactions on a particle-particle level. To solve this issue, FLIP and PIC use grids to simplify the problem. Individually, these solutions have their drawbacks but they rely on the same core calculations. Therefore, they can be combined to produce a realistic representation of liquid fluids.

Author's address: Joey Marten, McGill University, joey.marten@mail.mcgill.ca.

2023. XXXX-XXXX/2023/4-ART $15.00
https://doi.org/0000001.0000001

2 RELATED WORK

Alternatively, non-grid approaches, such as SPH Roy [1995], exist. On the other hand, pure Eulerian grid methods are common in the film and video game industries such as Stable Fluids designed by Stam [2003] however, this paper focuses on the previously mentioned methods. The core functions described here are based on the work done by Zhu and Bridson [2005] on Animating Sand as a Fluid. Additional implementation details were provided by Englesson et al. [2011] and Müller [2022] from which this paper takes inspiration.

3 METHODS

In order to understand the following implementation, some background is needed. The Navier-Stokes equations describe the fundamental workings of fluids.

$$\frac{\partial V}{\partial t} = F + \nu\nabla^2 V - V \cdot \nabla V - \frac{\nabla p}{\rho} \tag{1}$$

$$\nabla \cdot V = 0 \tag{2}$$

The first equation describes the change in a velocity field over time. F, the external force, accounts for gravity and the like. $\nu\nabla^2 V$ is the viscosity term which can be expressed as the diffusion of velocity. $V \cdot \nabla V$ is interpreted as the advection of velocity. Lastly, the pressure gradient is $\frac{\nabla p}{\rho}$ which is scaled by the density. The second equation outlines the divergence-free condition.

The algorithm can be broken down into the following procedures:

(1) Initialize the grid and particles
(2) Transfer the particle velocity to the grid and make a copy
(3) Solve incompressibility constraint
(4) Calculate PIC and FLIP velocities and transfer back to particles
(5) Update the particle positions using an ODE solver.
(6) Repeat steps 2-5

3.1 Particles

The particles are handled collectively by a particle system script. Each individual particle has properties for position, velocity, and mass. Additional properties are used for visualization but are not critical. After the velocities are calculated, the particle positions are updated using symplectic Euler integration. Other methods could be used to achieve similar effects. Lastly, particles can find themselves within the walls of the simulation. This is handled by pushing them out in the normal direction. Further collision response could be done to make the movement more realistic.

3.2 MAC Grid

The MAC grid is a variation of the traditional grid method. Instead of storing information in the center of the grid, velocities are stored

along the boundaries. Horizontal velocities are positioned along vertical boundaries while vertical velocities are stored on the horizontal boundaries.

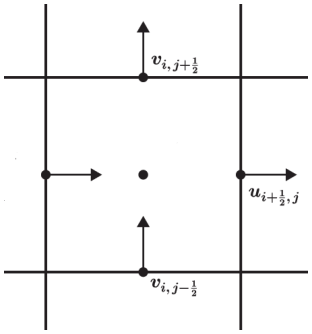

Fig. 1. The Marker and Cell method for the cell on the ith column and jth row

3.3 Transferring velocities to the grid

Each time step starts with the velocity transfer to the grid. For both the vertical and horizontal velocities, each particle adds a weighted velocity to the 4 closest grid velocities. The weight is calculated in a similar way to bilinear interpolation. Müller [2022] describes the method for indexing the velocity arrays. It involves subtracting a term (depending on whether the desired velocity is horizontal or vertical) from the x and y components of the particle. Afterwards, the result is divided by the cell size and floored to obtain the index for the stored values.

Furthermore, each cell is assigned a type. Cells containing a particle are set to fluid. A separate array contains the positions of all the solid cells. All other cells are saved as air cells. At this point, a copy of the velocities is made in order to calculate the FLIP velocities later on. Lastly, solid cells are processed to enforce the Dirichlet boundary condition which states that velocity along the normal from a non-solid cell into a solid cell should be zero. This can be stated as:

$$V \cdot n = 0 \tag{3}$$

Additionally, the implementation presented treats the boundaries as frictionless. The velocities orthogonal to the normal direction remain unaffected.

3.4 Solving divergence

Once the grid velocities are initialized, we begin to approximate a solution for equation 2. Divergence can be seen as the change of density in the cell over time. This means that a negative divergence results in particles being accumulated in the cell over time. In contrast, a positive divergence would give rise to cells that expel particles over time. For water-like fluids, we want the total divergence to be zero so that the overall density on the grid remains the same. Thankfully, the Marker And Cell method provides a simple way to compute divergence. We use the following formula:

$$div = u_{i+1,j} - u_{i,j} + v_{i,j+1} - v_{i,j} \tag{4}$$

Using the MAC grid (figure 1) as a blueprint, we calculate the divergence as the difference of flows out of and into the cell as stipulated in Müller [2022]. We see that if all the velocities are pointing out of the cells (ie $u_{i+1,j}, v_{i,j+1} > 0$ and $u_{i,j}, v_{i,j} < 0$) then the divergence will be positive as we stipulated above. We can solve this linear system using an iterative solver. For this implementation, I opted to use Gauss-Seidel for its simplicity but the preconditioned Conjugate Gradient method would be preferred. One adjustment must be made to account for solid boundaries. Flow can't cross a boundary in the normal direction and thus the divergence must reflect this. An additional check is made for the cell type. The following procedure is applied to each cell and repeated for k iterations:

(1) $div \leftarrow u_{i+1,j} - u_{i,j} + v_{i,j+1} - v_{i,j}$
(2) $s \leftarrow s_{i-1,j} + s_{i+1,j} + s_{i,j-1} + s_{i+1,j+1}$
    - where $s_{i,j} = 0$ if the cell (i,j) is a solid and = 1 otherwise
(3) $u_{i,j} \leftarrow u_{i,j} + div * s_{i-1,j}/s$
(4) $u_{i+1,j} \leftarrow u_{i+1,j} - div * s_{i+1,j}/s$
(5) $v_{i,j} \leftarrow v_{i,j} + div * s_{i,j-1}/s$
(6) $v_{i,j+1} \leftarrow v_{i,j+1} - div * s_{i,j+1}/s$

3.5 Transferring velocities to particles

To transfer the velocities from the grid, two velocities are calculated for each particle. One uses the PIC method while the other uses FLIP. They are then combined to a final velocity using a scalar ratio. This operation is summarized by the following equation:

$$v_{particle} = \alpha * v_{PIC} + (1 - \alpha) * v_{FLIP}, \ where\ \alpha \in [0, 1] \tag{5}$$

An alpha of 0 results in the complete PIC method while an alpha of 1 produces the FLIP solution.

3.5.1 PIC. The PIC method of velocities, as Zhu and Bridson [2005] describe, involves taking the result of a bilinear interpolation for the 4 closest grid values for each component. In 2D, the interpolation is performed twice, once for the horizontal velocities and once for the vertical portion. The PIC method suffers from numerical dissipation which results in a more viscous-looking fluid. Unfortunately, the technique fails to account for the particle's initial velocity. Particles close to each other end up with very similar velocities and thus they stay together.

3.5.2 FLIP. The FLIP method calculates the velocity by adding a component to the particle's existing velocity as detailed in Zhu and Bridson [2005]. The difference is taken between the velocities after the divergence solver and the copy of the velocities saved before. Then for each particle, the component is determined through a bilinear interpolation of the change in velocities.

$$\Delta v = v_{div-free} - v_{copy}$$

$$v_{FLIP} = v_{particle} + BilinearInterpolation(pos_{particle}, \Delta v)$$

4 RESULTS

The results can be separated into two parts: visual fidelity and compute performance. The overall visuals are quite convincing. Although it is highly subjective, it was found that a PIC/FLIP ratio of 0.01 produced a convincing image. When using entirely PIC driven velocities, the fluid clumps up and lacks the randomness of real water. In contrast, the purely FLIP simulation lacks cohesion

between particles especially on the surface of the fluid where it meets the air. It is worth noting that these algorithms are susceptible to numerical drift. One obvious sign of this is the convergence of particles. There is no collision handling thus as horizontal velocities get smaller, particle positions tend to overlap.

Here are the results using a 40x30 grid and 10000 particles with a cell size of 0.2.

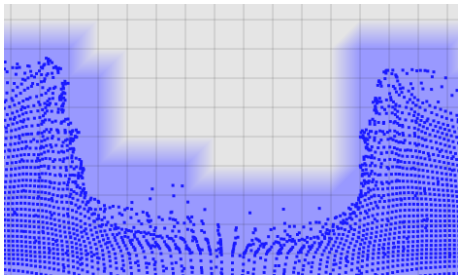

Fig. 2. FLIP droplet test

We see that the pure FLIP fluid scatters particles. Individually, they are not influenced too much by the neighboring particles. The fluid becomes grainy, especially at the surface.

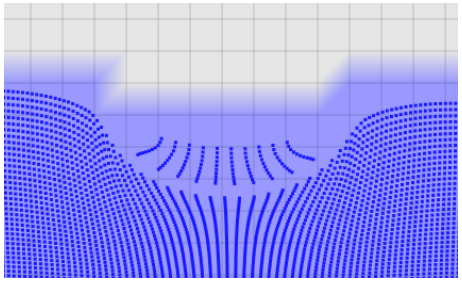

Fig. 3. PIC droplet test

The pure PIC has noticeable numerical dissipation which causes the fluid to form more uniformly. In turn, this results in visuals similar to a liquid with high viscosity.

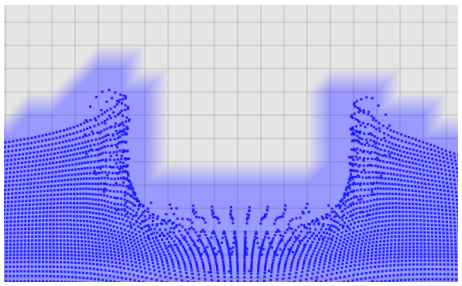

Fig. 4. Droplet test using PIC/FLIP ratio = 0.01

Lastly, using a ratio of 0.01 PIC to FLIP results in a fluid that retains some viscosity while introducing natural-looking velocity to particles in isolation. The surface particles remain free while the overall fluid keeps its shape.

To test the efficiency of the algorithm, two test systems were designed. The first is similar to a dam break while the second drops 1% of the total particles into a basin of the remaining particles. The tests were performed on two sizes of grids 80x60 and 40x30. The total number of particles varied as well with values of 5000 and 10000. The following parameters were used on all the tests: dt = 0.01 sec, dx = 0.2, PIC/FLIP ratio = 0.01, and 20 iterations for the incompressibility solver. Lastly, the tests were performed on a Windows 10 system with an Intel Core 1.20 GHz M7-6Y75 dual-core processor and 16GB of ram.

Dam Break (in seconds)		
Grid size and # of particles	5000	10000
40x30	27.462	33.633
80x60	30.798	38.645

Droplet (in seconds)		
Grid size and # of particles	5000	10000
40x30	27.103	31.554
80x60	30.134	34.422

I believe the simulation performs well given the processor. We see that the number of particles plays a larger role in overall compute time. By doubling the number of particles, the compute time increases by 22.47% in the first test and 16.43% in the second. On the other hand, quadrupling the number of cells causes the compute time to increase by 12.15% and 11.18% respectively. It's important to note that the tests account for the rendering of the system which impacts the total compute time. Further improvements could be made to leverage multi-threading and parallel computing on a GPU.

5 CONCLUSIONS

In this paper we have introduced and implemented the FLIP/PIC method for animating fluids. The concepts behind the algorithm are taken from both the Lagrangian and Eulerian points of view. Using MAC grids to store cell velocity greatly simplifies calculations of divergence in addition to streamlining boundary conditions. The performance results proved that particles have a worse effect on compute time compared to an increase in the number of cells. Nonetheless, it proved that it could be an effective solution to pre-rendered fluids on low-end devices. This method could likely be used for real-time applications with more advanced hardware and GPU acceleration. Visually, the FLIP and PIC methods produce satisfactory results but they are limited to the types of fluids being portrayed. Combining them leads to a more versatile outcome that can be used to emulate realistic water.

REFERENCES

D. Englesson, J. Kilby, and J. Ek. 2011. Fluid Simulation Using Implicit Particles. (12 2011). http://www.danenglesson.com/images/portfolio/FLIP/rapport.pdf

M. Müller. 2022. How to write a FLIP Water Simulation. (12 2022). https://matthias-research.github.io/pages/tenMinutePhysics/18-flip.pdf

T. Roy. 1995. *Physically Based Fluid Modeling Using Smoothed Particle Hydrodynamics*. University of Illinois at Chicago. https://books.google.ca/books?id=6WpONwAACAAJ

J. Stam. 2003. Real-Time Fluid Dynamics for Games. (05 2003). https://www.researchgate.net/publication/2560062_Real-Time_Fluid_Dynamics_for_Games

Y. Zhu and R. Bridson. 2005. Animating Sand as a Fluid. *ACM Trans. Graph.* 24, 3 (jul 2005), 965–972. https://doi.org/10.1145/1073204.1073298