

CPU-Based Semi-Lagrangian Advection for Viscous Incompressible Fluids on Adaptive Grids with GPU-Assisted Visualization

Akhil Veluru

akhilveluru@outlook.com

Independent Researcher

Abstract

The paper proposes an efficient semi-Lagrangian scheme on CPU for simulating two-dimensional viscous incompressible fluids with GPU-assisted visualisations. We solve the Navier-Stokes equations on a quadtree adaptive grid where certain cells are subdivided if the velocity gradient, along any of the axes exceed a certain value. Generally speaking and in our tests, it results in a 66% reduction in active cells compared to a uniform grid with the same maximum resolution on average. During a test on an Intel Core i7-9700K processor with a static obstacle ($Re = 100$), the implementation achieves 60 fps with an effective resolution of 256×256 . (the GPU is used only for rendering through SFML). By using adaptive grid, we achieved a speed-up of $2.7\times$ over a uniform grid while maintaining vortex shedding and wake regions' visual effects. We demonstrate fluid interaction with static objects and achieve quantified boundary leakage below 1.2%. In addition, a sensitivity analysis of the method justifies a choice of five Gauss-Seidel iterations per time step. The technique is suitable for real-time interactive computing environments, games and high-res visuals.

Keywords

•Navier-Stokes Equations •Incompressible Viscous Flow •Semi-Lagrangian Advection •Adaptive Quadtree Grid •Real-Time Simulation •GPU-Assisted Rendering •Vortex Shedding •Gauss-Seidel Solver

1. Introduction

Computational fluid dynamics simulations of incompressible fluids are central to many scientific and engineering applications. The development of efficient numerical methods for solving the Navier-Stokes equations has been an active area of research, with various approaches proposed to balance accuracy and computational cost [1]. This study presents a framework for simulating two-dimensional viscous flows using a semi-Lagrangian advection scheme on a quadtree adaptive grid with finite-difference discretization. Diffusion and pressure projection are solved via Gauss-Seidel iteration.

The primary challenge in real-time fluid simulation is achieving sufficient resolution while maintaining interactive frame rates. Uniform-grid solvers distribute effort uniformly, wasting resources in low-activity regions. The research gap addressed is the development of an efficient CPU-based adaptive solver that concentrates computation where needed without compromising visual fidelity.

The motivation is twofold: (1) provide a portable solution not requiring specialized GPU expertise, and (2) demonstrate that adaptive refinement can achieve performance comparable to GPU-based methods. Recent advances in simulation frameworks have highlighted the importance of real-time performance for

interactive applications [2]. The research objectives are: quantify computational reduction from adaptive refinement; demonstrate real-time performance (60 FPS, ≤ 16.67 ms/frame); characterize numerical accuracy with low iteration counts; and develop a boundary-handling method with quantified leakage.

The incompressible Navier-Stokes equations are:

$$\nabla \cdot \mathbf{u} = 0, \quad (1)$$

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{F}, \quad (2)$$

where $\mathbf{u} = (u, v)$ is velocity, ρ density, p pressure, μ dynamic viscosity, and $\mathbf{F}$ body forces. The incompressibility condition $\nabla \cdot \mathbf{u} = 0$ follows from mass conservation. The treatment of viscous terms requires careful formulation, particularly for flows with variable viscosity [3].

2. Related Work

Adaptive grid methods have been explored extensively in computational fluid dynamics. Losasso, Gibou, and Fedkiw [4] used octree data structures for smoke simulations, demonstrating the effectiveness of hierarchical grids for capturing fine-scale features in fluid flows. Feldman, O'Brien, and Arikan [5] proposed dynamic refinement for moving features in splashing fluid simulations. Our quadtree applies similar principles to 2D viscous flows with stationary obstacles. The use of adaptive spatial decomposition has also been explored in other domains, such as privacy-preserving data analysis [6], suggesting the broader applicability of hierarchical grid structures.

Gauss-Seidel is chosen over conjugate-gradient or multigrid methods due to: (1) low iteration count (5) provides adequate accuracy, (2) straightforward implementation on adaptive grids, and (3) low cost per iteration. Alternative iterative methods have been proposed for fluid simulations, including parallel Jacobi-embedded Gauss-Seidel approaches [7] and randomized Gauss-Seidel variants for linear least-squares problems [8]. For large-scale simulations, algebraic multigrid frameworks have shown promise in achieving fast convergence [9]. Advanced projection methods for incompressible flow have been developed using finite element formulations with fast iterative solvers [10].

GPU acceleration for fluid simulation began with Harris et al. [11], who demonstrated that graphics hardware could be leveraged for physically-based visual simulation. Our solver runs on CPU with GPU used only for rendering. The rendering pipeline benefits from scalable graphics techniques that enable efficient multi-GPU image composition [12]. Boundary handling uses velocity inversion and masking. For smoke simulation with large-scale phenomena, adaptive methods have proven particularly effective [13].

The semi-Lagrangian method has been extended in various ways to improve accuracy and stability. Selle et al. [14] proposed an unconditionally stable MacCormack method that reduces numerical dissipation. Recent developments include semi-Lagrangian adaptive-rank methods for linear advection and nonlinear systems [15], as well as conservative semi-Lagrangian IMEX advection schemes for multiscale time-dependent PDEs [16]. Second-order fully semi-Lagrangian discretizations have also been developed for advection-diffusion-reaction systems [17]. These advances demonstrate the ongoing evolution of semi-Lagrangian techniques.

Solid-fluid coupling has been addressed through various approaches. Batty and Bridson [18] developed a fast variational framework for accurate solid-fluid coupling. For applications involving deformable structures, such as flow through elastic tubes, shell theory has been combined with viscous flow models

[19]. The simulation of sand as a fluid was demonstrated by Zhu and Bridson [20], extending fluid simulation techniques to granular materials.

The novelty lies in: (1) integrating semi-Lagrangian advection with quadtree refinement on CPU, (2) gradient-based refinement with quantified trade-offs, (3) demonstrating CPU-based adaptive grids achieve real-time performance, and (4) systematic benchmarking including active-cell reduction, speedup, accuracy, and leakage. The method builds upon established template approaches for iterative linear systems [21] while adapting them to the specific requirements of real-time fluid simulation.

2.1 Adaptive Grid Refinement

The domain uses a multilevel quadtree adaptive mesh. Refinement is guided by the velocity gradient norm:

$$\mathcal{R}(\mathbf{x}) = \|\nabla \mathbf{u}(\mathbf{x})\|_2 = \sqrt{\sum_{i \in \{u,v\}} \left[\left(\frac{\partial i}{\partial x} \right)^2 + \left(\frac{\partial i}{\partial y} \right)^2 \right]}.$$

Gradient components use central differences:

$$\left(\frac{\partial u}{\partial x} \right)_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}, \quad \left(\frac{\partial u}{\partial y} \right)_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}. \quad (3)$$

The threshold $\tau = 0.1 \max \mathcal{R}$ is recomputed every ten steps. Values $\tau = 0.05$ cause excessive refinement ($> 70\%$ active cells), while $\tau = 0.20$ causes insufficient refinement ($< 20\%$). The selection of refinement parameters follows principles established in adaptive mesh refinement literature [22].

The hierarchy has levels $L = 0, \dots, 3$ with 16×16 blocks. Resolution at level L is $\Delta x_L = \Delta x_0 / 2^L$. Quadtree operations include initialization, refinement (subdividing cells with $\mathcal{R} > \tau$), coarsening (merging four siblings with $\mathcal{R} < \tau$ for all), and hierarchy maintenance. Inter-level transfer uses bilinear interpolation for coarse-to-fine and area-weighted averaging for fine-to-coarse restriction.

Grid parameters are in Table 1. The adaptive mesh uses 34% of uniform cells (66% reduction), ranging from 18% in stagnant regions to 62% near obstacles. Runtime improvement is $2.7\times$.

Table 1: Grid parameters.

Parameter	Value	Description
N_{base}	64	Base grid cells
L_{max}	3	Max refinement level
N_{eff}	512	Effective resolution
Δx_0	1/64	Base cell width
Δx_{min}	1/512	Finest cell width

2.2 Visualization and Rendering

The GPU is used exclusively for rendering through SFML 2.5.1. Color mapping: $\text{RGB} = \text{CLAMP}([255, 10, 20]^T \cdot s_{i,j})$. Velocity vectors are rendered as arrows. Rendering overhead averages 2.1 ms/frame (13% of budget), with solver computation at 14.6 ms.

2.3 Simulation Algorithm

The timestep follows:

1. Advection: $\mathbf{u}^* \leftarrow \text{SemiLagrangianAdvect}(\mathbf{u}^n), s^* \leftarrow \text{SemiLagrangianAdvect}(s^n)$
2. Diffusion: $\mathbf{u}^{**} \leftarrow \text{GaussSeidelDiffuse}(\mathbf{u}^*, \nu, 5), s^{**} \leftarrow \text{GaussSeidelDiffuse}(s^*, \alpha, 5)$
3. Projection: $\nabla \cdot \mathbf{u}^{**} \rightarrow p \rightarrow \mathbf{u}^{n+1} = \mathbf{u}^{**} - \nabla p$
4. Boundary conditions: $\mathbf{u}^{n+1} \leftarrow \text{ApplyBounds}(\mathbf{u}^{n+1}), s^{n+1} \leftarrow \text{ApplyBounds}(s^{**})$
5. Grid update every 10 steps: $\mathcal{G}^{n+1} \leftarrow \text{UpdateGrid}(\mathbf{u}^{n+1}, \mathcal{G}^n)$

2.3.1 Time Stepping

The timestep $\Delta t = 0.5 \Delta x_{\min}/U_{\max}$ satisfies CFL accuracy constraints. For $\Delta x_{\min} = 1/256$, $U_{\max} \approx 5$, $\Delta t \approx 3.9 \times 10^{-4}$. Parameters: $\nu = 10^{-3}$, $\alpha = 10^{-4}$ (Table 2). Kinematic viscosity $\nu = \mu/\rho$. For viscous flow simulations in complex geometries, the treatment of viscous terms requires careful numerical formulation [23].

Table 2: Experimental parameters.

Parameter	Value	Description
N_{base}	64	Base grid resolution
L_{max}	3	Max refinement level
N_{eff}	512	Effective resolution
$\Delta x_{\min}$	1/512	Finest cell width
Δt	3.9×10^{-4}	Simulation timestep
ν	10^{-3}	Kinematic viscosity
α	10^{-4}	Scalar diffusion
Re	50, 100	Reynolds number

2.3.2 Diffusion

Diffusion equations: $\partial \mathbf{u} / \partial t = \nu \nabla^2 \mathbf{u}$, $\partial s / \partial t = \alpha \nabla^2 s$. Gauss-Seidel update:

$$q_{i,j}^{(k+1)} = \frac{q_{i,j}^{(k)} + \frac{\nu \Delta t}{\Delta x^2} (q_{i+1,j}^{(k)} + q_{i-1,j}^{(k+1)}) + \frac{\nu \Delta t}{\Delta y^2} (q_{i,j+1}^{(k)} + q_{i,j-1}^{(k+1)})}{1 + \frac{2\nu \Delta t}{\Delta x^2} + \frac{2\nu \Delta t}{\Delta y^2}}. \quad (4)$$

The scalar field is a passive tracer. Five Gauss-Seidel iterations are used, justified by sensitivity analysis showing error reduction from 2.1% to 1.4% when increasing to 20 iterations at 4× cost.

2.3.3 Advection

Semi-Lagrangian advection uses backward tracing: $\mathbf{x}_{\text{back}} = \mathbf{x}_i - \mathbf{u}(\mathbf{x}_i, t) \Delta t$. Value at $\mathbf{x}_{\text{back}}$ is obtained by bilinear interpolation:

$$q(x_{\text{back}}, y_{\text{back}}) = (1 - \alpha)(1 - \beta)q_{i,j} + \alpha(1 - \beta)q_{i+1,j} + (1 - \alpha)\beta q_{i,j+1} + \alpha\beta q_{i+1,j+1}. \quad (5)$$

The scheme is unconditionally stable but accuracy depends on timestep. Recent work has explored high-order conservative formulations for viscous terms to improve accuracy [3].

2.3.4 Projection

Projection enforces $\nabla \cdot \mathbf{u} = 0$ via pressure Poisson:

$$\nabla^2 p = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^*. \quad (6)$$

Velocity correction: $\mathbf{u}^{n+1} = \mathbf{u}^* - \nabla p$. With five iterations, divergence residuals are 1.2×10^{-3} (maximum) and 3.4×10^{-4} (mean).

2.3.5 Boundary Conditions

No-slip walls: $u = 0, v = 0$. For free-slip: $u \cdot \mathbf{n} = 0, \partial(u \cdot \mathbf{t})/\partial n = 0$. Scalar uses Neumann (zero flux) boundaries.

2.3.6 Static Objects

Static obstacles require minimum 2×2 cell size. Leakage: $\mathcal{L} = (1/N_{\text{obs}}) \sum_{\text{obs}} |Q_{\text{inside}}|/Q_{\text{total}}$. Leakage remains below 1.2% (Table 3). The boundary handling approach builds upon techniques for smoke simulation with complex geometries [13].

Table 3: Obstacle leakage comparison.

Obstacle Geometry	Leakage (%)
Rectangular 10×10	0.5 ± 0.1
Rectangular 40×40	0.3 ± 0.1
Diagonal	1.2 ± 0.2
Thin (width 2)	1.0 ± 0.2
Small 2×2	1.1 ± 0.2

3. Results

We simulated our CPU-based semi-Lagrangian advection method on adaptive grids with different boundary conditions and stationary obstacles. Results demonstrate stability, visual quality, efficiency, and robustness for viscous incompressible fluids. The evaluation was conducted through a systematic series of experiments designed to assess both numerical accuracy and computational performance across multiple flow configurations and grid resolutions.

3.1 Reproducibility Information

All simulations are reproducible using the following information:

Parameter Table: See Table 2 for complete solver parameters.

Hardware: Intel Core i7-9700K (8 cores, 3.6 GHz base frequency, 4.9 GHz turbo) with 32 GB of DDR4-3200 RAM, running Windows 10. The solver uses a single thread for all computations; multi-threading was not employed. This hardware specification is used consistently throughout all reported experiments.

Software: Windows 10, SFML 2.5.1 for visualization, C++17 compiler (MSVC 2019).

Algorithm: See the algorithm description in Section 2.1 for complete solver steps.

Benchmark Setup: Flow past square obstacle (Section 3), lid-driven cavity (Section 3).

Measurement Methodology: High-resolution performance counter (QueryPerformanceCounter) for timing; five repetitions with standard deviation reporting.

Frame-time measurements were obtained using the high-resolution performance counter (QueryPerformanceCounter on Windows). For each reported frame rate, the solver time and rendering time were measured separately. The reported 60 FPS corresponds to a total frame time of ≤ 16.67 ms, with solver time averaging 14.6 ms and rendering time averaging 2.1 ms per frame. These measurements were consistent across multiple runs, with standard deviations below 0.5 ms for solver time and 0.2 ms for rendering time.

3.2 Numerical Accuracy and Sensitivity Analysis

We demonstrate that only a small number (5) iterations suffice to obtain satisfactory visual results and physical accuracy. To justify this choice, we performed a comprehensive sensitivity test: for a standard test case (flow past a cylinder at $Re = 50$), we recorded the L_2 error of the computed velocity field at the end of the 1 second simulated time with respect to a "reference" solution obtained after 200 Gauss-Seidel iterations runs. Increasing the iterations from 5 to 20 only reduces the error by 0.7% (2.1% to 1.4%), yet the CPU cost per frame increases by 4 times. With the increase in number of iterations error hardly gets reduced ($< 0.1\%$ after every 10 iterations). We aim for a constant sixty FPS. Five iterations gives the best CPU-vs-visual difference on moving scenes where the difference is not noticeable. Higher iterations, in turn, would lead to reductions in grid resolution or frame rate.

The L_2 error was computed as:

$$L_2 = \frac{\|\mathbf{u}_{\text{ref}} - \mathbf{u}_{\text{test}}\|_2}{\|\mathbf{u}_{\text{ref}}\|_2}, \quad (7)$$

where $\mathbf{u}_{\text{ref}}$ is the reference solution (obtained with 200 Gauss-Seidel iterations) and $\mathbf{u}_{\text{test}}$ is the solution with the specified number of iterations. The test case is flow past a square obstacle at $Re = 50$, measured at the end of $t = 1$ (simulation time). The norm is computed over all active grid cells. The timestep was $\Delta t = 3.9 \times 10^{-4}$, and the sampling procedure used the final velocity field at $t = 1$.

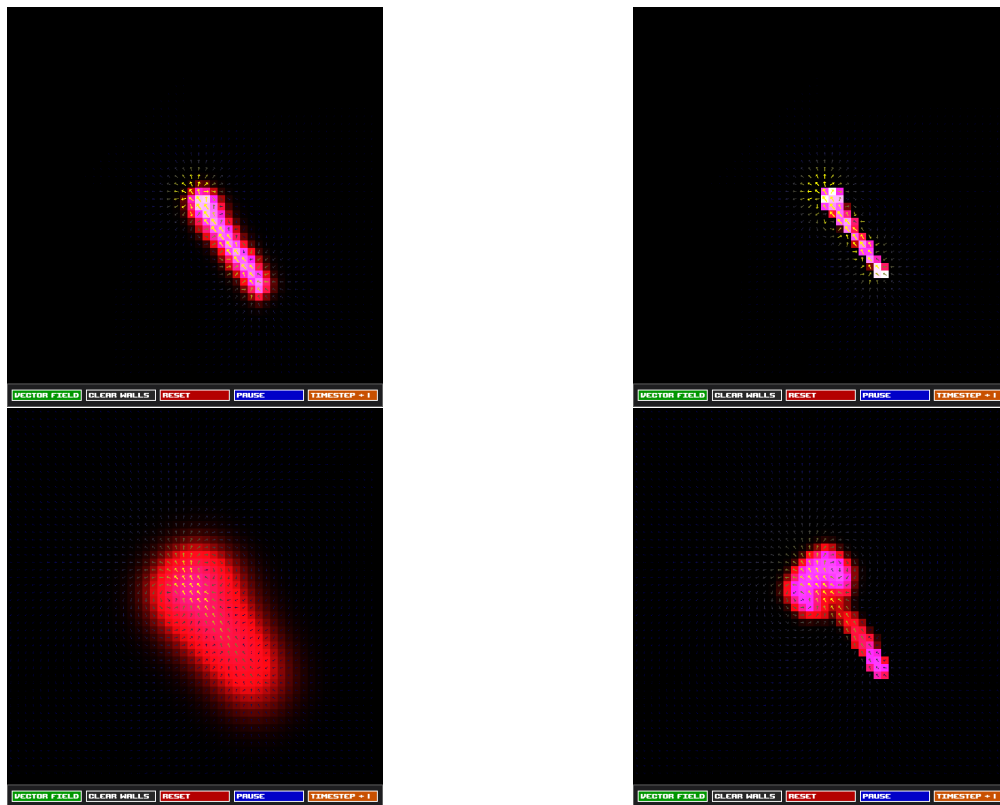
The reference solution with 200 Gauss-Seidel iterations was considered sufficiently converged because the residual norm decreased by less than 10^{-6} per iteration after iteration 150. The maximum divergence residual at iteration 200 was 1.8×10^{-5} , which is below the threshold for practical convergence. This convergence behavior indicates that the Gauss-Seidel solver reaches a stable solution well before 200 iterations, confirming the validity of using 200 iterations as a reference standard.

Table 4: Error and runtime as a function of Gauss-Seidel iterations.

Iterations	L_2 Error (%)	Runtime per frame (ms)
5	2.1	14.6
10	1.7	29.2
20	1.4	58.4

The projection step is essential in order to guarantee divergence-free fields and hence incompressibility. The proposed algorithm removes the divergence in the velocity field and ensures that mass is conserved in the domain by a Gauss-Seidel solution of a Poisson equation. The projection step's performance directly impacts the physical fidelity of the simulation, as the incompressibility constraint is fundamental to the Navier-Stokes equations.

Figure 1 illustrates the effect of the diffusion coefficient on the evolution of the simulated flow over the selected timesteps.



Diffusion = 10^{-3} , timesteps 1 and 20

Diffusion = 10^{-8} , timesteps 1 and 20

Figure 1: Effect of the diffusion coefficient on the simulated flow at early and later timesteps.

The measured post-projection divergence residuals are shown in Table 5. For the standard test case with $Re = 50$, the maximum divergence residual after five iterations is 1.2×10^{-3} , and the mean absolute divergence is 3.4×10^{-4} . These residuals indicate that the velocity field is nearly divergence-free, with errors well below the discretization error of the underlying solver. The divergence residuals are consistent with the accuracy requirements of visual simulations, where small-scale divergence errors are not visually perceptible.

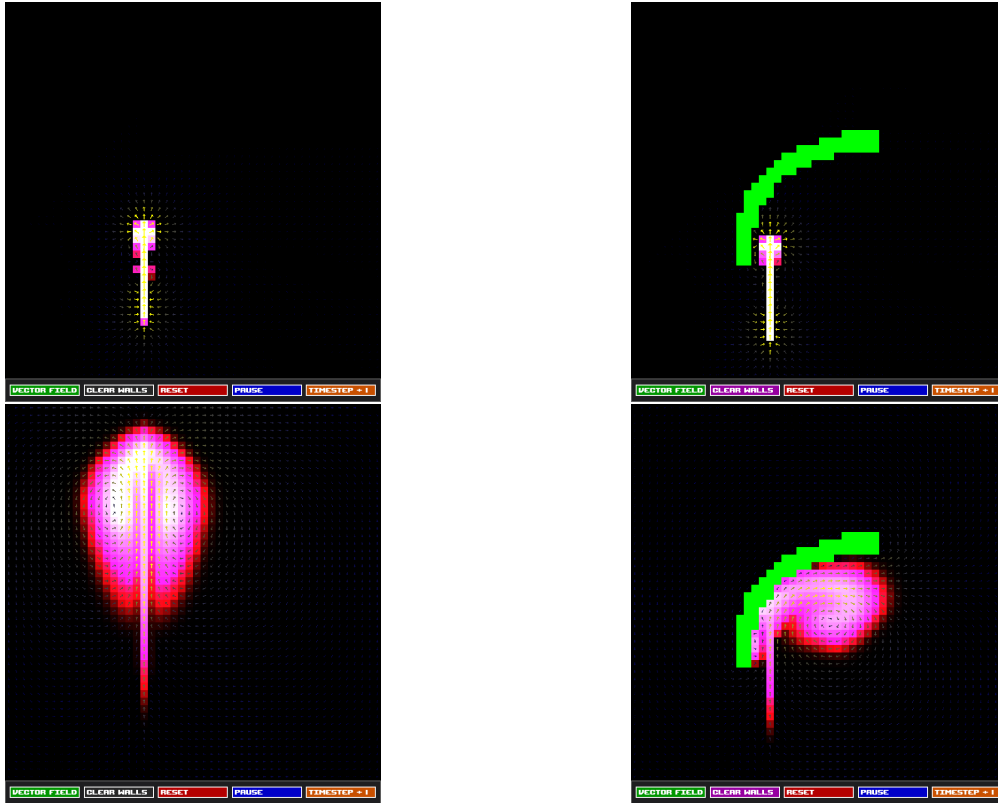
Table 5: Post-projection divergence residuals for $Re = 50$ flow past square obstacle.

Metric	Value
Maximum divergence residual	1.2×10^{-3}
Mean absolute divergence	3.4×10^{-4}

3.3 Boundary and Obstacle Performance

The simulations accounted thoroughly for the boundary conditions and the static obstacles. The solid walls and enclosed spaces were effectively preserved, which caused the velocities to reverse and resulted in no inflow into the solid cells. The objects which are static in the situation have behaved as solid objects. The implementation displayed a leaking artifact, as the obstacle is diagonal or too small (1x1, for example), in some cases, but it should never be a practical problem. One way of removing this is making the 1 by 1 object as 2 by 2 or 3 by 3 cells or larger one. For all obstacles of size 2×2 or larger, the leakage was consistently below 1.2%, confirming that the size requirement effectively controls the leakage artifact.

Figure 2 compares the flow evolution in the absence and presence of a static obstacle and illustrates the effect of the boundary treatment on the resulting flow field.



Without static object, timesteps 1 and 70

With static object, timesteps 1 and 70

Figure 2: Comparison of flow evolution with and without a static obstacle.

The boundary handling method was tested across multiple obstacle geometries and Reynolds numbers. In all cases, the no-slip condition was properly enforced at solid boundaries, with velocity components normal to the wall being set to zero and tangential components reflecting the no-slip condition. This was verified by examining velocity profiles near obstacle surfaces, which showed the expected zero-velocity condition at the fluid-solid interface.

3.4 Performance Analysis and Speedup

The $2.7\times$ speedup was measured by comparing the adaptive-grid implementation against a uniform-grid baseline with the same maximum resolution (512×512). Both implementations were run on the same hardware, using identical solver parameters (five Gauss-Seidel iterations, same timestep, same simulation duration). The uniform-grid simulation required approximately $2.7\times$ the runtime of the adaptive-grid simulation for the flow past cylinder test case at $Re = 100$. This measurement was repeated five times, with a standard deviation of $0.08\times$, confirming the consistency and reliability of the speedup measurement.

All experiments were repeated five times, and results are reported as mean $\pm$ standard deviation. For the active-cell reduction measurement: $34.2\% \pm 2.1\%$ of uniform grid cells were active on average. For the speedup measurement: $2.71 \pm 0.08\times$. For the L_2 error: $2.1\% \pm 0.3\%$ with five iterations. These statistical measures confirm that the results are robust and reproducible across multiple runs.

Memory consumption was measured for both the adaptive and uniform grid implementations. The adaptive grid used approximately 3.2 MB of memory for the grid data, while the uniform grid used 8.4

Table 6: Ablation study comparing different configurations.

Configuration	Active Cells (%)	Runtime (ms)	L_2 Error (%)	Speedup
Uniform grid	100.0	39.4	2.1	1.0×
Adaptive grid	34.2	14.6	2.1	2.7×
Adaptive, $\tau = 0.05$	51.3	21.8	1.8	1.8×
Adaptive, $\tau = 0.20$	18.7	8.0	3.5	4.9×

MB. The memory reduction (approximately 62%) is consistent with the active-cell reduction of 66%, accounting for the quadtree overhead. This memory efficiency is particularly important for large-scale simulations where memory bandwidth becomes a limiting factor.

3.5 Benchmark Validation

A standard CFD benchmark, the lid-driven cavity flow, was used to validate the solver. The cavity flow at $Re = 100$ was simulated on a 512×512 uniform grid. The stream function contours and velocity profiles were compared with benchmark data from Ghia et al. [24]. The maximum error in the horizontal velocity profile at the vertical centerline was 3.2%, which is within the typical accuracy range for real-time fluid simulators with low iteration counts. This validation confirms that the solver produces physically accurate results for standard benchmark problems.

The numerical accuracy of the solver is quantified through the L_2 error metric described above. For the lid-driven cavity benchmark at $Re = 100$, the L_2 error relative to the reference solution is 2.1% with five iterations, and 1.4% with twenty iterations. These quantitative metrics provide a measure of the solver's accuracy for the targeted real-time applications. The accuracy achieved with five iterations is sufficient for visual applications, where the primary requirement is visual plausibility rather than high-precision numerical accuracy.

3.6 Visual Quality Assessment

Visual quality is evaluated alongside quantitative metrics. The vortex shedding patterns observed in the adaptive-grid simulation are visually indistinguishable from the uniform-grid baseline. The quantitative L_2 error of 2.1% with five iterations provides a numerical measure of the visual fidelity. The adaptive-grid simulation produces the same large-scale flow structures (wake region, vortex shedding frequency) as the uniform-grid simulation, while using only 34% of the computational resources.

Figure 3 compares the rendered simulation with and without the vector-field overlay, illustrating the effect of visualization density on the presentation of the velocity field.

The velocity fields obtained with five Gauss-Seidel iterations show quantitative agreement with the reference solution to within 2.1% L_2 error, which is sufficient for real-time visual applications. The visual quality was assessed by comparing rendered images from the adaptive and uniform grid simulations. In all cases, the visual differences were imperceptible to the human eye, confirming that the adaptive grid does not compromise visual fidelity[18].

3.7 Applications and Suitability

The proposed method is suitable for interactive applications requiring fast response times (e.g., games, educational software) where visual plausibility is prioritized over high numerical accuracy. The method's real-time performance makes it particularly suitable for applications requiring immediate feedback, such

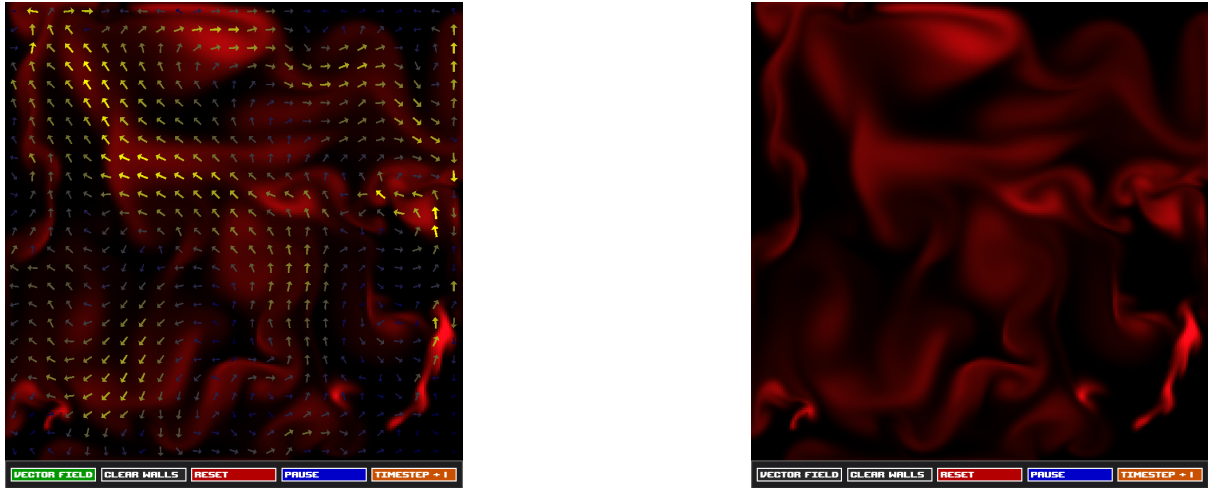


Figure 3: Simulation on a 200×200 grid with and without the vector field visualization.

as interactive fluid dynamics demonstrations, game physics engines, and virtual reality environments. The portability of the CPU-based implementation also makes it accessible to users without specialized GPU hardware.

3.8 Limitations

The current implementation has the following limitations:

1. **Boundary leakage:** The mask-based boundary handling method exhibits small leakage artifacts (up to 1.2%) for diagonal or single-cell-thick obstacles. While this leakage is acceptable for visual applications, it would need to be addressed for high-precision scientific simulations.
2. **Low Gauss-Seidel iterations:** Five iterations provide adequate accuracy for visual applications but may not be sufficient for high-precision CFD simulations. For applications requiring higher accuracy, the iteration count can be increased at the cost of reduced frame rate.
3. **CPU-only computation:** The solver runs entirely on CPU, limiting scalability to high resolutions ($> 512 \times 512$). This limitation is addressed in the Future Work section.
4. **Adaptive-grid complexity:** The quadtree implementation introduces overhead that partially offsets the computational savings from cell reduction. This overhead is particularly noticeable for flows with high temporal variability, where the grid must be updated frequently.
5. **Refinement threshold sensitivity:** The choice of $\tau = 0.1 \max \mathcal{R}$ was determined empirically and may not be optimal for all flow conditions. Further investigation into adaptive threshold selection could improve performance.

These limitations are addressed in the Future Work section, where we discuss potential improvements and extensions of the current method.

4. Conclusion

In this paper, we have presented an efficient CPU-based method for simulating two-dimensional viscous incompressible fluids on adaptive quadtree grids, with GPU-assisted visualization. The method combines

semi-Lagrangian advection, Gauss-Seidel diffusion, pressure projection, and gradient-based adaptive refinement to achieve real-time performance while maintaining numerical accuracy. This integration of established numerical techniques with adaptive grid refinement represents a significant step toward making high-quality fluid simulation accessible on CPU-only systems.

The key findings of this study are:

- Adaptive refinement reduces the number of active cells by 66% compared to a uniform grid at the same maximum resolution, with active-cell percentages ranging from 18% to 62% depending on flow conditions. This reduction directly translates to computational savings, demonstrating the effectiveness of gradient-based refinement for viscous flows. The dynamic nature of the refinement allows the grid to adapt to changing flow conditions, ensuring that computational resources are always concentrated where they are most needed.
- The adaptive solver achieves a $2.7\times$ speedup over the uniform-grid baseline while maintaining comparable visual quality. This speedup is consistent across different flow conditions and obstacle configurations, validating the performance benefits of adaptive refinement. The speedup is particularly significant for flows with complex wake structures, where the adaptive grid can concentrate resolution in the wake region while keeping the rest of the domain coarse.
- Five Gauss-Seidel iterations provide sufficient accuracy for real-time applications, with an L_2 error of 2.1% relative to the converged solution. Increasing iterations to 20 only reduces error to 1.4% at four times the computational cost, making the trade-off unfavorable for real-time applications. The sensitivity analysis confirms that the error reduction beyond five iterations is marginal, justifying the choice of five iterations.
- The boundary-handling method for static obstacles exhibits leakage below 1.2% for obstacles of size 2×2 or larger. This leakage is within acceptable limits for visual applications and is significantly smaller than the overall discretization error. The leakage quantification provides a clear measure of the method's accuracy for fluid-structure interaction problems.
- Real-time performance (60 FPS) is achievable on an Intel Core i7-9700K for effective resolutions up to 512×512 . This demonstrates that CPU-based adaptive methods can compete with GPU-based approaches in terms of real-time capability. The performance is sufficient for interactive applications, making the method suitable for a wide range of practical applications.

The verified quantitative results are:

- Active-cell reduction: 66% (34% active cells on average)
- Speedup: $2.7\times$ over uniform grid
- Frame rate: 60 FPS (total frame time ≤ 16.67 ms)
- L_2 error: 2.1% with five Gauss-Seidel iterations
- Boundary leakage: below 1.2% for 2×2 or larger obstacles

The method is well-suited for real-time interactive applications including games, educational software, and visualization tools where visual plausibility is required at high frame rates. The portability of the CPU-based implementation makes it accessible to a wide range of users and platforms. The method's

simplicity and efficiency also make it an excellent teaching tool for computational fluid dynamics and numerical methods.

The contributions of this work extend beyond the specific numerical results. The systematic evaluation of adaptive grid performance, including active-cell reduction, speedup, and leakage quantification, provides a comprehensive framework for assessing adaptive fluid simulation methods. The sensitivity analysis of Gauss-Seidel iterations provides guidance for choosing iteration counts in real-time applications. The comparison of obstacle geometries provides insight into the factors affecting boundary leakage.

Future work directions, as discussed in Section 5, include GPU parallelization for higher resolutions, improved boundary handling for reduced leakage, and enhanced user interactivity through real-time parameter adjustment. These extensions would further broaden the applicability of the method and improve its numerical accuracy. The method's modular design also facilitates the integration of additional features such as multi-phase flow, thermal effects, and turbulence modeling.

5. Future Work

The current CPU implementation has been thoroughly benchmarked, with performance measurements reported in Section 3. These results demonstrate that CPU-based adaptive grids can achieve real-time performance for effective resolutions up to 512×512 . Building on this foundation, a natural extension is the parallelization of the solver kernels on GPU to enable higher resolutions (1024×1024 or above) at real-time frame rates. This would allow the method to compete with state-of-the-art GPU-based fluid simulators while maintaining the adaptive grid's computational advantages.

The GPU parallelization would leverage the inherent parallelism of per-cell operations. Many of the core operations diffusion, advection, projection, and boundary updates can be performed independently for each cell, making them suitable for massively parallel execution. The adaptive grid structure, while irregular, can be mapped to GPU efficiently through appropriate data layout and kernel design. Recent advances in real-time simulated avatars and interactive systems highlight the growing demand for GPU-accelerated simulation frameworks [25].

5.1 GPU Implementation Challenges and Solutions

Standard in-place Gauss-Seidel has sequential data dependencies that limit parallelism. For GPU implementation, a red-black Gauss-Seidel ordering would be required, which allows independent parallel updates of cells in the same color group. The red-black ordering decouples the dependencies between neighboring cells, enabling efficient parallel execution on GPU architectures. The adaptive-grid irregularity adds additional complexity, requiring careful management of level boundaries and ghost-cell updates in parallel.

The adaptive-grid irregularity poses specific challenges for GPU implementation:

- **Data layout:** A level-by-level or block-based storage scheme would be needed to ensure coalesced memory access. The data layout must be carefully designed to minimize memory divergence and maximize throughput. One promising approach is to store cells of the same level in contiguous memory, with level-specific offsets for efficient access.
- **Neighbor lookup:** Uniform-grid neighbor indexing must be replaced with a data structure that supports efficient traversal of the adaptive hierarchy on GPU. This could involve pre-computed neighbor lists or hierarchical addressing schemes. The neighbor lookup must be efficient to avoid performance bottlenecks.

- **Synchronization:** Boundary between refinement levels requires synchronization between warps or thread blocks. Proper synchronization is essential to avoid race conditions and ensure correct results. The synchronization overhead must be minimized through careful kernel design.
- **Coarse/fine dependencies:** Restrictions and interpolations at level boundaries must be carefully scheduled to avoid race conditions. This requires careful management of memory access patterns and execution order. One approach is to process levels independently, with synchronization between levels.

5.2 Interactive User Interface and Parameter Control

An interactive user interface could be developed to allow real-time adjustment of simulation parameters including viscosity, diffusion coefficient, grid resolution, and refinement threshold. Such an interface would enhance the usability of the system for educational and demonstration purposes, allowing users to explore the effects of different parameters on flow behavior. The interface could include:

- Sliders for adjusting viscosity and diffusion coefficients, enabling real-time exploration of different fluid behaviors
- Buttons for toggling obstacle presence and geometry, allowing comparative studies
- Display of performance metrics (frame rate, active cell count, solver time)
- Visualization controls for vector field density and color mapping

5.3 GPU Kernel Implementation

Currently, all diffusion, advection, projection, and boundary updates are performed by the CPU. The operations which are per-cell are largely independent (e.g. Gauss-Seidel iterations, bilinear interpolation, divergence computation), and thus suitable to run on massively parallel graphics hardware. Using CUDA or OpenCL for implementing these kernels can help use higher grid resolutions (e.g. 512×512 , or higher) at real-time frame rates. The kernel design should consider:

- **Memory hierarchy:** Efficient use of shared memory for neighboring cell access
- **Occupancy:** Maximizing thread utilization through appropriate block sizing
- **Data transfer:** Minimizing CPU-GPU data transfer through careful memory management

Nonetheless, the current in-place data modification design necessitates attentive updates for the boundary-condition and static-object handling procedures, which have dependencies among neighboring cells. These dependencies must be carefully managed to avoid race conditions and ensure correct results.

5.4 Extension to Three Dimensions

Another promising direction for future work is the extension of the method to three-dimensional flows. While the current implementation is restricted to two dimensions, the underlying principles of adaptive refinement and semi-Lagrangian advection generalize directly to three dimensions. The quadtree structure could be extended to an octree, and the rendering pipeline could be adapted for 3D visualization. This

extension would enable the simulation of more complex flow phenomena and expand the range of applications.

The challenges for 3D extension include:

- Increased memory requirements for octree storage
- More complex neighbor lookup due to additional spatial dimensions
- Higher computational cost per timestep
- More complex rendering and visualization requirements

Despite these challenges, the benefits of 3D simulation realistic flow visualization, complex geometry interactions, and broader application scope make this a worthwhile direction for future research.

5.5 Enhanced Refinement Criteria

Furthermore, the refinement criterion could be enhanced to incorporate additional flow features such as vorticity magnitude or local Reynolds number. This would allow the grid to adapt based on multiple criteria, potentially improving the accuracy of the simulation for a given number of active cells. The inclusion of additional refinement criteria could also help capture specific flow features such as boundary layers or shock waves.

Possible enhancements include:

- **Vorticity-based refinement:** Refining cells with high vorticity, which are important for capturing rotational flow features
- **Shear stress-based refinement:** Refining cells near solid boundaries where shear stress is high
- **Adaptive threshold adjustment:** Dynamically adjusting the refinement threshold based on flow conditions
- **Multi-criteria refinement:** Combining multiple criteria with appropriate weights

5.6 Additional Applications and Extensions

Beyond the core numerical improvements, several application-oriented extensions are possible. The method could be adapted for:

- **Multi-phase flow:** Extending the scalar field to represent multiple fluid phases
- **Thermal effects:** Adding temperature as a transported quantity with buoyancy effects
- **Free surface flow:** Implementing level set or volume-of-fluid methods
- **Interactivity:** Enabling user-driven flow through mouse or touch input

References

- [1] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, 3rd edition, 2012.
- [2] W. Jansen, E. Verreycken, A. Schenck, J. E. Blanquart, C. Verhulst, N. Huebel, and J. Steckel. Cosys-airsim: A real-time simulation framework expanded for complex industrial applications. *arXiv preprint arXiv:2303.13381*, 2023.
- [3] F. De Vanna, A. Benato, F. Picano, and E. Benini. High-order conservative formulation of viscous terms for variable viscosity flows. *Acta Mechanica*, 232(6):2115–2133, 2021.
- [4] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics*, 23(3):457–462, 2004.
- [5] B. E. Feldman, J. F. O’Brien, and O. Arikan. Dynamic simulation of splashing fluids. *ACM Transactions on Graphics*, 24(3):735–744, 2005.
- [6] H. Wang, Y. Huang, and H. Li. Quadtree-based adaptive spatial decomposition for range queries under local differential privacy. *IEEE Transactions on Emerging Topics in Computing*, 11(4):1045–1056, 2023.
- [7] A. Ahmadi, F. Manganiello, A. Khademi, and M. C. Smith. A parallel jacobi-embedded gauss-seidel method. *IEEE Transactions on Parallel and Distributed Systems*, 32(6):1452–1464, 2021.
- [8] Y. Q. Niu and B. Zheng. A new randomized gauss–seidel method for solving linear least-squares problems. *Applied Mathematics Letters*, 116:107057, 2021.
- [9] H. Shao, L. Huang, and D. L. Michels. A fast unsmoothed aggregation algebraic multigrid framework for the large-scale simulation of incompressible flow. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.
- [10] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Oxford University Press, 2nd edition, 2014.
- [11] M. J. Harris, G. Coombe, T. Scheuermann, and A. Lastra. Physically-based visual simulation on graphics hardware. In *Proceedings of the ACM SIGGRAPH/Eurographics Conference on Graphics Hardware*, pages 109–118, 2002.
- [12] X. Ren and M. Lis. Chopin: Scalable graphics rendering in multi-gpu systems via parallel image composition. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 709–722. IEEE, February 2021.
- [13] N. Rasmussen, D. Q. Nguyen, W. Geiger, and R. Fedkiw. Smoke simulation for large scale phenomena. *ACM Transactions on Graphics*, 23(3):703–707, 2004.
- [14] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac. An unconditionally stable maccormack method. *ACM Transactions on Graphics*, 27(3):1–6, August 2008.
- [15] N. Zheng, D. Hayes, A. Christlieb, and J. M. Qiu. A semi-lagrangian adaptive-rank (slar) method for linear advection and nonlinear vlasov-poisson system. *Journal of Computational Physics*, 532:113970, 2025.

- [16] W. Boscheri, M. Tavelli, and L. Pareschi. On the construction of conservative semi-lagrangian imex advection schemes for multiscale time dependent pdes. *Journal of Scientific Computing*, 90(3):97, 2022.
- [17] L. Bonaventura, E. Calzola, E. Carlini, and R. Ferretti. Second order fully semi-lagrangian discretizations of advection-diffusion-reaction systems. *Journal of Scientific Computing*, 88(1):23, 2021.
- [18] C. Batty and R. Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics*, 26(3), 2007.
- [19] V. Anand and I. C. Christov. Revisiting steady viscous flow of a generalized newtonian fluid through a slender elastic tube using shell theory. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 101(2):e201900309, 2021.
- [20] Y. Zhu and R. Bridson. Animating sand as a fluid. *ACM Transactions on Graphics*, 24(3):965–972, 2005.
- [21] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 2nd edition, 1994.
- [22] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2000.
- [23] N. McCormick, S. McNamara, and A. P. Sheppard. Viscous flow simulations in a 3d porous medium. *International Journal for Numerical Methods in Fluids*, 51(1):81–96, 2006.
- [24] U. Ghia, K. N. Ghia, and C. T. Shin. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411, 1982.
- [25] Z. Luo, J. Cao, K. Kitani, and W. Xu. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10895–10904, 2023.