Adaptive Distributed Simulation of Fluids and Rigid Bodies

Haoyang Shi
University of Utah
USA
Roblox
USA
haoyang.shi@utah.edu

Victor Zordan
Roblox
USA
vbzordan@roblox.com

Yin Yang
University of Utah
USA
yin.yang@utah.edu

Sheldon Andrews
École de technologie supérieure
Canada
Roblox
USA
sheldon.andrews@etsmtl.ca

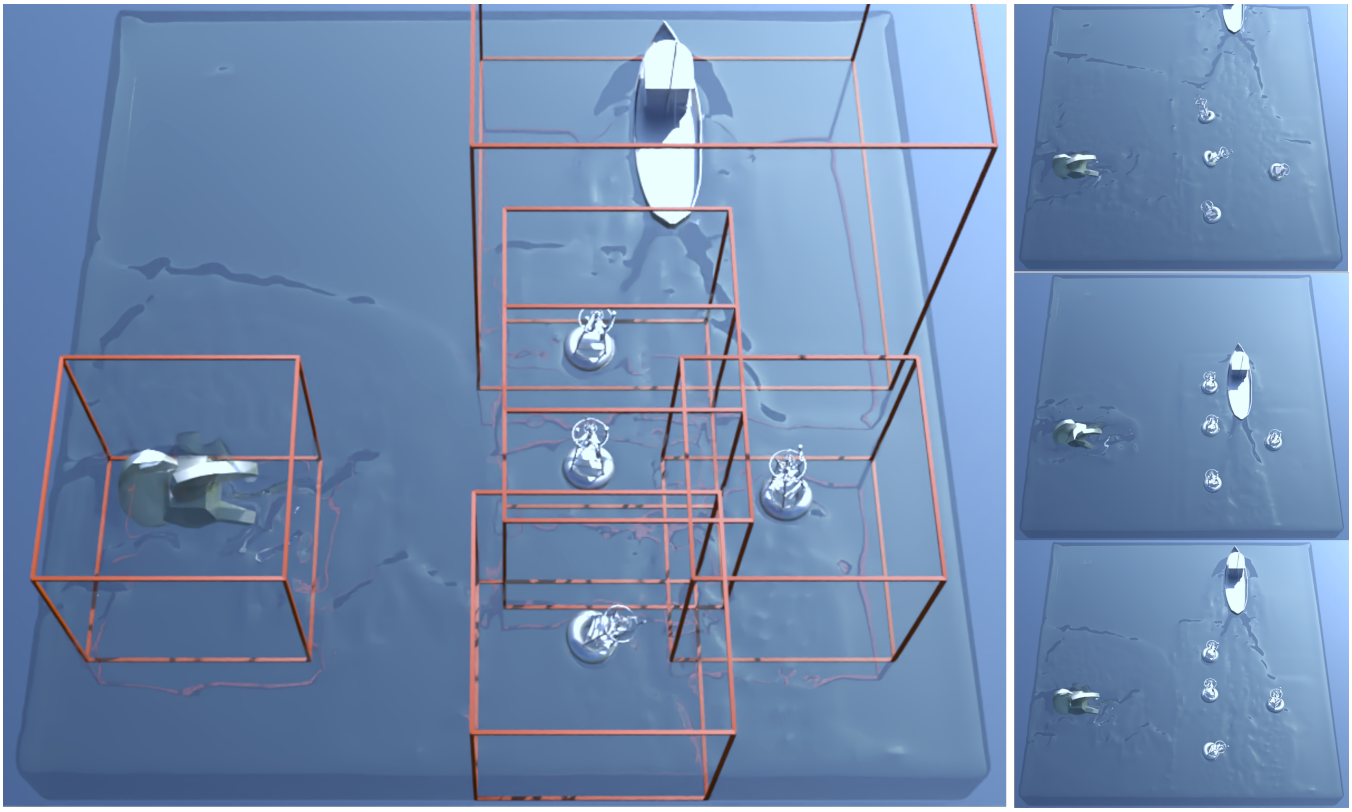


Figure 1: A boat moves past several buoys while a propeller spins in the water. The buoys experience the effects of both the boat wake and the larger scale effects of the propeller. Right column: show the dynamic behavior and rigid-fluid coupling using our adaptive multigrid scheme. Leftmost image: visualizes the local grids that refine the fluid simulation in regions surrounding each dynamic object, which are stepped at a higher framerate than the global grid.

MIG '24, November 21–23, 2024, Arlington, VA, USA

ABSTRACT

We present a framework for the interactive simulation of fluids coupled with rigid bodies that targets heterogeneous distributed computing architectures. Specifically, our proposed approach is well-suited for computer graphics applications that combine servers

with large compute capactiy with low-end devices. In this setting, a global large-scale fluid simulation is performed on servers using high-end compute hardware, and local refinement of fluid and rigid body coupling is performed on a client with limited compute resources, such as a tablet or smartphone. We demonstrate the effectiveness of our framework to simulate large and complex scenes involving wind, ocean, and dynamic objects, all while providing plausible interactions through fluid-rigid coupling.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; *Distributed simulation.*

KEYWORDS

fluids, distributed simulation, rigid bodies, physics-based animation

ACM Reference Format:

Haoyang Shi, Victor Zordan, Yin Yang, and Sheldon Andrews. 2024. Adaptive Distributed Simulation of Fluids and Rigid Bodies. In *The 17th ACM SIGGRAPH Conference on Motion, Interaction, and Games (MIG '24), November 21–23, 2024, Arlington, VA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3677388.3696334

1 INTRODUCTION

Fluid simulation plays an important role in many computer graphics applications, including special effects in films, engineering visualization, and video games. The demand for interactivity in the latter necessitates overcoming computational challenges, especially on mobile devices such as tablets, smartphones, and low-end laptops, where achieving real-time frame rates can be daunting. Despite these challenges, mobile devices are amongst the most popular class of devices for gaming. However, few works in computer graphics and physics-based animation have specifically aimed at developing methods for mobile devices, especially for fluid applications.

Next generation online games and VR applications demand heightened realism and interactivity, which includes simulating physical phenomena such as rigid bodies, collisions, and fluids. Executing these simulations on the local device is crucial since response time is important, and remote execution of simulations may introduce unwanted latency and unresponsiveness. However, the limited computing power of so-called thin clients prohibits the simulation of rich, complex environments using traditional approaches. Therefore, developing methods that address the problem of physics simulation in the described context is essential. This work addresses the requirements of applications with demand for coupled fluid and rigid body simulation in a distributed setting.

Specifically, we present a multigrid distributed simulation framework that assumes a common heterogeneous mixture of computing resources, combining thin clients with high-end servers. Our methodology employs an adaptive multigrid method, performing a global fluid simulated on a coarse grid computed on a server, and coupling it with a number of fine-scale simulation on local grids that are potentially executed on one or more clients. We employ an adaptive refinement strategy to the flow field obtained on the coarse grid for the higher-resolution client grids. We also propose a novel blending strategy that merges flow field information from the local grids back to the global simulation, offering details from the local regions of interests back to the coarser global context. Finally, we adopt a strategy where simulations are carried out at different time scales-— low-frequency phenomena is simulated with larger time steps, while higher-frequency details are computed at higher rates (60 Hz) consistent with rigid body simulations for game environments.

To validate the feasibility of our method, we showcase its application in several complex scenes. The results demonstrate the stability and efficiency of our proposed approach in handling fluid simulation coupled with rigid bodies.

2 RELATED WORK

Here we provide a brief review of prior work in fluid simulation in computer graphics and scientific computing. Given the extensive body of literature on this subject, our focus is on parallelized and distributed physics simulation, and the coupling of fluids with rigid bodies since these topics overlap with our proposed technique.

2.1 Parallel Fluid Simulation

Parallel algorithms are common for both particle and grid-based fluid simulations. In particle simulations, parallelization is typically achieved through local computation of forces, done iteratively to address global behavior.

For grid-based fluid simulations, various approaches exist. Iterative solvers, such as the Jacobi method and parallel implementations of sparse conjugate gradient, are useful for the diffusion and divergence computations and offer straightforward parallelization. Stam [1999] pioneered the semi-Lagrangian approach in graphics, advocating the use of a Jacobi-type fixed-point solver [Stam 2003]. Numerous variants of this approach have since emerged within the graphics community. In our work, a Gauss-Seidel smoother comprises our multigrid scheme, optimizing performance by leveraging multi-core hardware.

While many frameworks concentrate on utilizing local multicore hardware, e.g. Ament et al. [2010], our framework is designed for a heterogeneous distributed environment. This assumes an environment where certain devices possess substantial computational resources (e.g., servers or high-end graphics workstations), while others, such as tablets, have limited compute capabilities.

Previous graphic studies have explored domain decomposition techniques [Chu et al. 2017; Liu et al. 2016], which decomposes the simulation into smaller sub-domains plus the interface between them. Wang et al. [2020] proposed algorithmic and data structure optimizations to enhance the scalability of the material point method (MPM) in a multi-GPU setting. However, their focus is on achieving higher fidelity simulations, requiring minutes per frame, whereas we target interactive and real-time applications.

Others have explored real-time distributed physics simulation, although principally for rigid bodies with constraints [Brown et al. 2019; De Oliveira et al. 2024].

2.2 Scalable, Adaptive and Hybrid Fluid Solvers

Multigrid solvers [Trottenberg et al. 2000] are amongst the most efficient and scalable for fluid simulation. Geometric multigrid methods have had particular uptake in computer graphics due to the simplicity of implementing coarsening and refinement operators on

regular grid domains. [McAdams et al. 2010] used geometric multigrid as a preconditioner for the conjugate gradient (CG) method, which significantly improves performance and robustness. One of the advantages of geometric multigrid is that it is easily combined with spatial adaptivity [Ferstl et al. 2014; Liu et al. 2016], a feature that we leverage in our work. Chentanez and Müller [2011] explored the use of heightfields for efficient fluid simulation on GPUs.

Adaptive mesh-refinement (AMR) method uses a hierarchy of axis-aligned grid patches to improve resolution where required. This allows for a straightforward implementation, although it could require a large number of refinements if fine level of detail is required. Our approach resambles the Berger-Oliger-Colella adaptive refinement [Berger and Colella 1989; Berger and Oliger 1984]. However, a key difference is that our framework performs integration at different scales for the coarse and fine grids and we handle overlapping adaptive patches.

Chimera grids are a more general class of adaptive methods that allow for oriented grids, which increases the flexibility of refinement since grids can be aligned with boundary features. In graphics, English et al. [2013] demonstrate the benefits of Chimera grids for achieving efficient, yet detailed, simulations on distributed parallel hardware. However, Chimera grids require additional data structures, e.g. Voronoi mesh, to determine stencil information in overlapping regions, and automatic and robust methods for computing such data structures is a challenge.

The divergence-free computation has long been recognized as a bottleneck in grid-based fluid simulations. Earlier work in graphics has examined economizing the pressure projection work done on a coarse grid by mapping pressure values to a fine grid and then performing local pressure refinements [Lentine et al. 2010]. We are inspired by such approaches to reduce the computational load of clients with limited computing resources, and our framework goes one step further by amortizing the global divergence computation over larger time scales.

Maintaining large grid structures can increase the memory footprint requirements for simulation. Golas et al. [2012] propose a hybrid technique coupling a moving Eulerian domain and vortex domain to reduce the memory footprint of large-scale fluid simulations. The method demonstrates impressive speedup on a limited number of examples, yet performance improvements diminish for highly dynamic scenes, which are characteristic of games and other interactive applications. Our approach assumes a setting where large-scale storage and global computations are offloaded to high-end servers, and devices with limited computing resources are only responsible for refinement on limited regions near dynamic objects. Huang et al. [2021] aims to capture larger-scale wave effects combined with local fluid interactions, and propose a hybrid-solver that combines a local FLIP simulation with the boundary element method used to simulate the larger domain. Their ambitions are similar to our own, although they do not target our regime of applications.

2.3 Fluid-Rigid Coupling

Many works in computer graphics and scientific computing have addressed fluid-solid coupling. The seminal work by Foster and Metaxas [1996] demonstrated grid-aligned one-way coupling using a marker-and-cell (MAC) discretization. This requires a rasterization of the solid onto the grid and in order to determine boundary conditions. Our work focuses on two-way coupling between fluids and dynamic rigid bodies. Takahashi et al. [2002] achieve two-way coupling by assigning zero Neumann boundary conditions for pressures of any grid cell more than half filled with a solid. Cell velocities covered by a solid are then assign velocities from the solid, although this approach cannot couple forces and torques due to fluid momentum.

The immersed boundary method [Peskin 2002] is another popular technique that gives a continuous forcing function that allows the fluid pressure field to change the solid's velocity by considering the solid to be part of the fluid. The related approach by Carlson et al. [2004] fixes the solid density to be the same as the fluid and enforcing a rigidity constraint for velocities inside the solid using Lagrange multipliers. Guendelman et al. [2005] propose a solid-fluid coupling approach specialized for thin shells and cloth.

Cut-cell methods are another approach for computing interactions between fluids and solids. They are popular due to their simplicity and ability to encode fluxes due to geometric details smaller than the grid resolution, but without having to refine or change the grid structure. These approaches were first used in computer graphics by Roble et al. [2005]. The area of grid cell faces covered by the solid boundary is used to improve the accuracy of the divergence calculations. [Batty et al. 2007] later used partial grid cell overlap as part of their variational formulation of pressure projection.

3 SYSTEM OVERVIEW

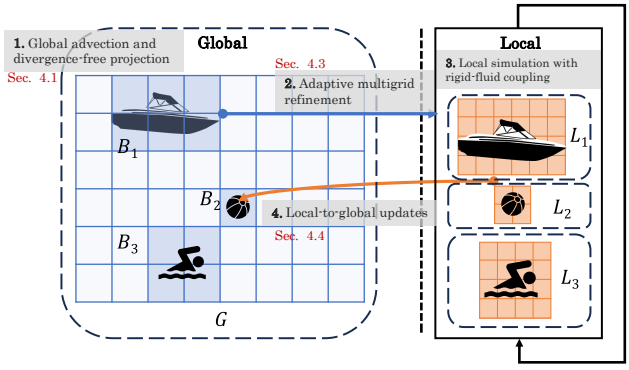


Figure 2: An overview of our adaptive multigrid simulation framework for fluid-rigid coupling targeting distributed heterogeneous computing platforms.

Our framework employs a client-server architecture where a coarse global simulation occurs on the server, while refined local simulations take place on one or more clients. The global simulation utilizes a coarse grid spanning the entire scene, and uses a larger time step for integrating velocities and pressures. In contrast, each local simulation uses a refined grid with higher resolution, focusing on a small region of the scene centered around one or more rigid bodies and using a smaller time step. The motivation for a multiscale paradigm is to give consistent low-frequency behavior

at the global scale, while enabling higher-frequency detail and responsive coupling at the local level. A conceptual visualization of our proposed framework is shown in Fig. 2, and the principal steps of our methodology are summarized below.

(1) **Global advection and divergence-free projection.** On the server, the fluid is simulated on a coarse grid using a semi-Lagrangian approach and time step of $\Delta t_G = N\,\Delta t$. Fluid-rigid coupling is achieved using the immersed boundary method. We employ a multigrid-preconditioned conjugate gradient (MGPCG) solver that extends the general method proposed by McAdams et al. [2010] to compute a divergence-free flow field on the coarse grid.

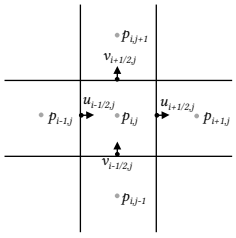
(2) **Adaptive multigrid refinement.** Next, an adaptive refinement strategy is used to refine the flow field solution obtained on the coarse grid by using a higher resolution grid centered at each rigid body group. Boundary information and flow field for grid cells occupied by dynamic objects are identified and forwarded to each client handling the refined simulation.

(3) **Local simulation with rigid-fluid coupling.** A stable semi-Lagrangian integration scheme and the immersed boundary method are again used to compute the updated flow field for the local grid, this time using the local time step Δt. This process is repeated N times, following which a restriction operator maps flow field values from the fine-to-coarse grid.
4. Local-to-global updates. Finally, updated flow field information is sent back to the server. In cases where multiple clients produce updated values for the same coarse grid cells, i.e. they overlap, the information from each client is merged using a novel and pragmatic blending scheme before integration.

4 METHODOLOGY

Our fluid simulation framework is based on the semi-Lagrangian approach developed by Stam [1999] for solving the Navier-Stokes equations. However, a significant difference is our use of a composite grid consisting of a coarse global grid G and multiple local grids $\{L_1, \ldots, L_K\}$ that potentially overlap. Each local grid L_k is paired with collection of one or more rigid bodies with constraints B_k.

The grids store the flow field velocities $\mathbf{u}$ and pressures $\mathbf{p}$ used by the fluid simulation. Sub-scripts $\cdot_G$ and $\cdot_{L_k}$ are used to distinguish between global and local quantities, respectively. Similarly, we store the generalized coordinates and velocities of rigid bodies $\mathbf{q}$ and $\dot{\mathbf{q}}$, respectively, and pressure forces $\mathbf{f}$ coming from the fluid simulation. The sub-script $\cdot_{B_k}$ is used to refer to a specific rigid body group.



Velocity and pressure values are stored using a staggered grid data structure. That is, pressure is stored at the center of grid cells, and velocity at the faces between cells. The inline image shows a 2D example of a staggered grid. We use the standard convention that splits the velocity into Cartesian components, such that u, v, and w are the horizontal, vertical and depth directions, respectively.

Algorithm 1: High-level algorithm of our distributed fluid-rigid simulation pipeline.

G = global grid
$L = \{L_1, L_2, \ldots L_K\}$ `// local grids`
$B = \{B1, B_2, ..B_K\}$ `// rigid body groups`
function `step()`:
  `// global sim`
  $\mathbf{u}'_G \leftarrow \text{ADVECT}(\Delta t_G, \mathbf{u}^t_G)$
  $\mathbf{f}_B \leftarrow \text{COMPUTEPRESSUREFORCES}(\Delta t_G, B, \mathbf{u}'_G)$
  $\mathbf{q}'_B, \dot{\mathbf{q}}'_B \leftarrow \text{INTEGRATE}(\mathbf{q}^{(t)}_B, \dot{\mathbf{q}}^{(t)}_B, \mathbf{f}_B)$
  $\mathbf{u}'_G, \mathbf{p}'_G \leftarrow \text{PROJECT}(\mathbf{u}'_G, \mathbf{q}'_B, \dot{\mathbf{q}}'_B)$
  `// in parallel`
  for $k = 1 \ldots K$ **do**
    `// local sim`
    for $i = 1 \ldots N$ **do**
      $\mathbf{u}'_{L_k} \leftarrow \text{ADVECT}(\Delta t, \mathbf{u}^{t_{i-1}}_{L_k})$
      `// Section 4.3.2`
      $\alpha \leftarrow \frac{i}{N}$
      $\mathbf{p}'_{L_k}, \mathbf{p}'_{\partial L_k} \leftarrow \text{INTERPOLATE}(\alpha, \mathbf{p}^t_G, \mathbf{p}'_G)$
      $\text{SETDIRICHLET}(L_k, \mathbf{p}'_{\partial L_k})$
      $\mathbf{f}_{B_k} \leftarrow \text{COMPUTEPRESSUREFORCES}(\Delta t, \mathbf{q}^{t_{i-1}}_{B_k}, \dot{\mathbf{q}}^{t_{i-1}}_{B_k}, \mathbf{u}'_{L_k})$
      $\mathbf{q}^{(t+i\Delta t)}_{B_k}, \dot{\mathbf{q}}^{(t+i\Delta t)}_{B_k} \leftarrow \text{INTEGRATE}(\mathbf{q}^{t_{i-1}}_{B_k}, \dot{\mathbf{q}}^{t_{i-1}}_{B_k}, \mathbf{f}_{B_k})$
      $\mathbf{u}^{(t+i\Delta t)}_{L_k}, \mathbf{p}_{L_k} \leftarrow \text{PROJECT}(\mathbf{u}'_{L_k}, \mathbf{q}^{(t+i\Delta t)}_{B_k}, \dot{\mathbf{q}}^{(t+i\Delta t)}_{B_k})$
    end
  end
  `// Synchronize local and global fluids`
  $\mathbf{u}'_G \leftarrow \text{UNIFIEDVIEW}(\mathbf{u}'_G, \mathbf{u}_L)$ `// Section 4.4`
  $\mathbf{u}^{t+\Delta t_G}_G \leftarrow \text{PROJECT}(\mathbf{u}'_G)$
  $\mathbf{p}^{t+\Delta t_G}_G \leftarrow \mathbf{p}'_G$
  `// Send local rigid body state back to global`
  $\mathbf{q}^{t+\Delta t_G}_B \leftarrow \{\mathbf{q}^{t+N\Delta t}_{B_1}, \ldots, \mathbf{q}^{t+N\Delta t}_{B_K}\}$
  $\dot{\mathbf{q}}^{t+\Delta t_G}_B \leftarrow \{\dot{\mathbf{q}}^{t+N\Delta t}_{B_1}, \ldots, \dot{\mathbf{q}}^{t+N\Delta t}_{B_K}\}$

For instance, $u_{i+\frac{1}{2},j,k}$ gives the horizontal velocity between grid cells (i, j, k) and $(i + 1, j, k)$.

Pseudo-code for computing a single step that advances the state of the fluid and rigid bodies is presented in Algorithm 1. The sections below provide further technical details of our methodology.

4.1 Global Simulation

The global fluid simulation on grid G is updated using two major steps: an advection step ADVECT that updates velocity and scalar values due to the current fluid flow, followed by a modified divergence computation on irregular boundary geometry. Next, a pressure projection PROJECT is applied to the velocity field to maintain a divergence-free state.

Computing the divergence-free velocity field $\mathbf{u}_G$ requires solving a Poisson equation. However, since our goal is to couple the fluid and solid objects, we must correctly handle the Neumann boundary conditions at the fluid-solid interface.

4.1.1 Divergence Stencil. Using a second-order accurate central difference scheme, the divergence of a single cell in 2D is

$$\frac{u_{i+1/2,j} - u_{i-1/2,j} + v_{i,j+1/2} - v_{i,j-1/2}}{h},$$

which computes the flux of the fluid through the faces divided by the cell size h. We denote this discrete divergence operator as $\mathbf{D}_h$, which maps a bundle of staggered single-component velocity samples to cell-centered divergence on the grid.

With the boundary geometry taken into account, the flux of the total material in cut-cell consists of a fluid contribution and solid contribution. To get the cell fluid flux, we mask the face flux by the corresponding fluid face fraction $\varphi_{i+1/2,j}$, $\varphi_{i-1/2,j}$, $\varphi_{i,j+1/2}$, $\varphi_{i,j-1/2}$. This gives the modified divergence operator as $\mathbf{DL}$, where $\mathbf{L}$ is a diagonal matrix formed by the fluid face fractions. Similarly, the flux of the rigid body is masked by $1-\varphi_{i+1/2,j}$, $1-\varphi_{i-1/2,j}$, $1-\varphi_{i,j+1/2}$, $1-\varphi_{i,j-1/2}$, and the divergence of a mixed fluid-rigid domain is approximated by $\mathbf{D}\left(\mathbf{L}\mathbf{u}_{\text{fluid}} + (\mathbf{I}-\mathbf{L})\mathbf{u}_{\text{rigid}}\right)$. Note that the subscript h is dropped here since we assume the grid cell size is fixed.

4.1.2 Divergence-free Projection. The divergence-free condition dictates that the sum of fluid and solid volume should stay the same, in other words, the flux of total material entering through the cell faces, should equal the amount that is leaving.

The divergence-free condition on the Cartesian grid can be expressed as the following linear system

$$\mathbf{D}\left(\mathbf{L}\mathbf{u}_{\text{fluid}} + (\mathbf{I}-\mathbf{L})\mathbf{u}_{\text{rigid}}\right) = 0,$$

where $\mathbf{u}_{\text{fluid}}$ are the divergence-free velocities after applying the pressure $\mathbf{u}_{\text{fluid}} = \mathbf{u}' - \frac{\Delta t}{\rho}\mathbf{F}\mathbf{p}$, with $\mathbf{F}$ the discrete gradient operator, and $\mathbf{u}'$ the velocity field after advection. Moving the unknowns, i.e., the pressure field, to the left-hand side, gives:

$$\frac{\triangle t}{\rho}\mathbf{DLF}\mathbf{p} = \mathbf{DL}\mathbf{u}' + \mathbf{D}(\mathbf{I}-\mathbf{L})\mathbf{u}_{\text{rigid}}. \tag{1}$$

After solving for pressure, the last step is to apply the pressure gradient to the velocity field $\mathbf{u}'$ to get an updated $\mathbf{u}$, such that

$$\mathbf{u} \leftarrow \mathbf{u}' - \frac{\Delta t}{\rho}\mathbf{F}\mathbf{p}. \tag{2}$$

We use an occupancy data structure to determine if a grid cell is empty, contains fluid, or is inside a solid object. If the center of a cell is empty, we mark it as a cell with Dirichlet condition $p_{i,j} = 0$. Otherwise, we use a level set on the global or local grid to determine whether it is completely inside a rigid body. If not, we mark it as an unknown for the pressure solve.

For each pair of adjacent Dirichlet cell and interior cell marked with an unknown pressure, we follow the *ghost fluid* method [Gibou et al. 2002] to set up our Dirichlet boundary conditions. At its simplest, the ghost fluid method brings the boundary resolution to sub-cell level by estimating where the water-air interface lies between two grid cells and enforcing the condition $p = 0$ exactly at the interface. In contrast, the voxelized approach can only enforce the $p = 0$ condition on cell centers.

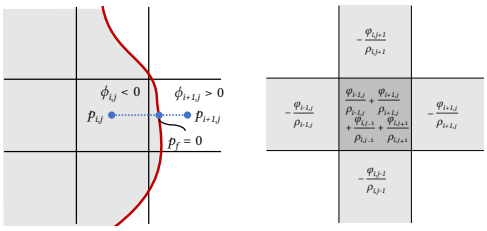


Figure 3: Left: Ghost pressures from a 2D grid are inserted at cell centers where $\phi_{i,j} > 0$. The boundary condition $p_f = 0$ is enforced at the position evaluated by the neighboring $\phi_{i,j}$ and $\phi_{i+1,j}$. Right: The Poisson stencil.

4.1.3 Dirichlet Boundary Conditions. We use the example shown in Fig.3 to explain the process. The pressure field is extrapolated into the adjacent empty cell using the pressure $p_{i,j}$ and $p_f = 0$ at the interface. The ghost pressure in this case is given by $p^{ghost}_{i+1,j} = \frac{\phi_{i+1,j}}{\phi_{i,j}} p_{i,j}$, which means we can express the ghost pressure as a linear combination of existing unknown, thus excluding them from the set of unknowns.

The end effect for the linear system in Equation 1 is scaling the diagonal entry corresponding to $p_{i,j}$ by $\frac{\phi_{i+1,j}-\phi_{i,j}}{\phi_{i,j}}$, and setting all the pressure values in empty adjacent grid cells to be 0 when we apply Eq. 2. Conceptually, this scaling factor is equivalent to the inverse of the average liquid density in the adjoint cell estimated on the two adjacent ϕ samples, denoted $\frac{1}{\rho}$. Following this idea, we construct the Poisson stencil shown in Fig. 3, which is expressed by the parameters φ and ρ.

Expressing the Poisson stencil with an explicit $\frac{1}{\rho}$ factor also gives us additional convenience, as the flexibility for changing the density will further facilitate fluid-solid coupling of rigid bodies of different densities.

4.1.4 Solid-Fluid Coupling. The pressure force acting on a rigid body may be computed by integrating the flux of the pressure gradient $\nabla\mathbf{p}$ over its volume V_{B_k}:

$$\mathbf{f}_{B_k} = \int_{V_{B_k}} \nabla\mathbf{p}\, dV. \tag{3}$$

Fedkiw [2002] notes that, for incompressible fluids, the pressure field for the divergence-free projection can be stiff and noisy, making it unsuitable for computing the coupling force that drives the solids. Therefore, a separate pressure field $\mathbf{p}_c$ is introduced to capture the reciprocated forces from the fluid.

We use the *immersed boundary method* [Peskin 2002] for solving $\mathbf{p}_c$. The force that the fluid exerts on the rigid body is calculated based on the difference between the rigid body motion and the fluid motion. In this scheme, the pressure forces are a function of the rigid body velocities $\dot{\mathbf{q}}_{B_k}$, which in turn is updated by the pressure force. By letting those two quantities be compatible, a fixed-point iterative method emerges (see Algorithm 2). Although multiple iterations

Algorithm 2: Computing pressure force

Data: $\mathbf{q}_{B_k}, \dot{\mathbf{q}}_{B_k}$ for rigid bodies in B_k, H grid (global or local)
Result: $\mathbf{f}_c$ pressure force

1 **function** `computePressureForces`$(\Delta t, \mathbf{q}_{B_k}, \dot{\mathbf{q}}_{B_k}, \mathbf{u}'_H)$:
2 $\quad \phi \leftarrow \min(\phi_{\text{rigid}}, \phi_{\text{fluid}})$
3 $\quad \mathbf{L} \leftarrow$ computeFaceFraction(ϕ) `// Section 4.1.1`
4 $\quad$ update volumes V_{B_k} from $\mathbf{q}_{B_k}$
5 $\quad \dot{\mathbf{q}}^{(0)} = \dot{\mathbf{q}}_{B_k}$
6 $\quad$ **for** $i = 1 \ldots$ `maxIter` **do**
7 $\quad\quad \mathbf{u}_{B_k} \leftarrow$ rigidToGrid$(H, \dot{\mathbf{q}}^{(i-1)})$
8 $\quad\quad$ solve $\frac{\triangle t}{\rho}\mathbf{D}\mathbf{f}\mathbf{p}_c^{(i)} = \mathbf{D}\mathbf{L}\mathbf{u}'_H + \mathbf{D}(\mathbf{I}-\mathbf{L})\mathbf{u}_{B_k}$ `// Eq. 1`
9 $\quad\quad \mathbf{f}_c^{(i)} \leftarrow$ integratePressure$(V_{B_k}, \mathbf{p}_c^{(i)})$ `// Eq. 3`
10 $\quad\quad \dot{\mathbf{q}}^{(i)} \leftarrow \dot{\mathbf{q}}_{B_k} + \Delta t \mathbf{f}_c^{(i)}$
11 $\quad\quad$ **if** $\|\dot{\mathbf{q}}^{(i)} - \dot{\mathbf{q}}^{(i-1)}\| < \epsilon$ **then**
12 $\quad\quad\quad$ **return** $\mathbf{f}_c^{(i)}$
13 $\quad\quad$ **end**
14 $\quad$ **end**

are helpful for refining the pressure forces, we set `maxIter` = 1 in practice and find that it typically gives a plausible solution. In the pseudocode, the RigidtoGrid function computes the rigid velocity on the staggered grid, and the integratePressure function integrates the pressure force according to Eq. 3.

4.2 Multigrid Solver

In this section, we describe the basic components of our geometric multigrid solver for solving the linear system in Eq. 1. The solver is based on the multigrid method proposed by McAdams et al. [2010], which we extend to work with non-voxelized geometry and spatially varying fluid density.

4.2.1 Smoother. We use the stencil described in Fig. 3 for smoothing. Specifically, 6 red-black Gauss-Seidel iterations are performed on the finest grid for pre-smoothing, and 6 for post-smoothing. The iteration count is doubled each time there is a transition to a coarser level, with a maximum cap of 24 iterations.

4.2.2 Grid Transfer. Transferring grid values from a finer to a coarser level consists of three parts:

- **Cell type.** A coarse grid cell is labeled as a "Dirichlet" cell if all of its children are also labeled as "Dirichlet". This is determined implicitly by checking if all children have 0 average density.
- **Restriction of the Poisson stencil.** The face fraction of a coarser grid can be computed as the average of its children weighted by the face area, which we denote as the face restriction operator R_f. Similarly, the average density of a coarse face cell can be averaged from the volume-weighted density of its children, denoted as density restriction operation R_ρ. In 2D, operators R_f and R_ρ are:

$$R_f = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}_h^{2h}, \qquad R_\rho = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix}_h^{2h}.$$

- **Restriction/Prolongation for approximations.** We use a constant restriction and tri-linear interpolation scheme for cell-centered multigrid. In 2D, the restriction I_h^{2h} and prolongation I_{2h}^h operators are:

$$I_h^{2h} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}_{2h}^{h}, \qquad I_{2h}^h = \frac{1}{16}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix}_h^{2h}.$$

Finally, we use a direct solver on the coarsest level.

4.3 Distributed Adaptive Refinement

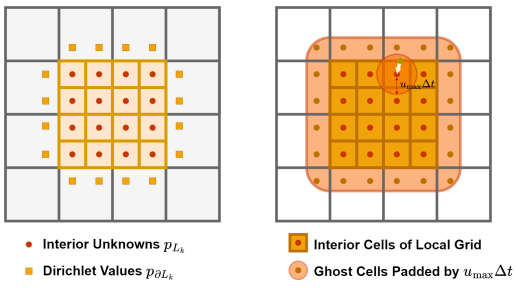
Adaptive refinement of the coarse grid allows high-frequency details and rigid-fluid coupling to be ameliorated by performing fine scale simulations in specific regions, e.g. the vicinity of rigid bodies. In a distributed setting, such refinements may be carried out remotely, and consequently the round trip time between compute nodes can easily become a bottleneck. Our design choice for adaptive refinement is therefore biased toward minimizing the dependence on the global grid.

Compared to the coarse grid simulation, the underlying Navier-Stokes equation and the advection-projection type stepping remain unchanged, and the only difference is we discretized on a higher resolution, plus we need to account for the boundary conditions imposed by the larger grid G surrounding the subdomains $L_{1\ldots K}$. This results in the addition of line 12-14 in Algorithm 1 to the local simulation compared to the global simulation. Moreover, the local projection uses a much simpler Gauss-Seidel solver as it starts from a fairly accurate initial guess given by the global pressure solve.

4.3.1 Local Advection. In the advection stage, a vicinity of radius $|\mathbf{u}_{\max}|\Delta t$ is explored as we backtrace from one grid point to its previous position. Therefore we interpolate the coarse fluid field on a band of width $|\mathbf{u}_{\max}|\Delta t$ ghost cells surrounding the local grid in addition to the interior grid points of the local grid.

4.3.2 Local Projection. After advection, the steps outlined in Section 4.1.1 are used to assemble the linear system Eq. 1 on local grid L_k. This requires computing a new modified boundary-aware divergence operator $\mathbf{DL}_{L_k}$ on the refined grid, and the total divergence of the advected refined flow field $\mathbf{DL}_{L_k}\mathbf{u}'_{L_k} + \mathbf{D}(\mathbf{I}-\mathbf{L}_{L_k})\mathbf{u}_{\text{rigid}}$.

As the coarse pressure is already known, we follow the coarse-to-fine full multigrid approach to solve Eq. 1. This starts by mapping the global pressures $\mathbf{p}'_G$ onto the local adaptive grid with tri-cubic interpolation, giving an initial approximation $\mathbf{p}'_{L_k}$. To apply the



correct compatible boundary condition, the interpolation domain is extended by 1 grid on the subdomain boundary, forming a 1-grid Dirichlet padding $\mathbf{p}_{\partial L_k}$ around the subdomain as shown below.

Based on the smoothing stencil $\mathbf{DL}_{L_k}$, initial solution $\mathbf{p}'_{L_k}$ and Dirichlet boundary pressure values $\mathbf{p}_{\partial L_k}$, 30 iterations of red-black Gauss-Seidel iterations are applied as the smoother, and we get an improved solution $\mathbf{p}_{L_k}$.

In a typical adaptive multigrid cycle, the coarse levels are revisited for at least another cycle. However, this would be expensive in a distributed setting since we assume that the fine (local) and coarse (global) grids are separated across the network. We therefore propose to use a single coarse-to-fine cycle, i.e. skip revisiting the coarser grids later on, and later use a special blending method to merge local flow field information back to the global grid.

Another notable difference of our approach is that multiple updates are performed on the local grid using an asynchronous substepping scheme. During intermediate substeps, the initial approximation of the pressure field and its boundary is interpolated from the temporal interpolation of the latest two available $\mathbf{p}'_G$.

Finally, the smoothed local grid pressure solution $\mathbf{p}_{L_k}$ stays on the local grid as we apply Eq. 2 to update the velocity field.

4.4 Synchronous Local-to-Global Updates

Due to asynchronously stepping the local grid, the flow field may begin to diverge at interfaces between the local and global grids. The global grid therefore must update its estimate of the flow field velocities $\mathbf{u}'_G$ by incorporating information from each of the local grids $\mathbf{u}_{L_{1\ldots K}}$. We refer to this step as *fluid blending*, which is then followed by a final Poisson computation to bring the flow field to a divergence-free state.

4.4.1 Fluid Blending. 3D fluid blending has not been a well-tackled research problem. To our knowledge, no scheme readily takes in two divergence-free fields and outputs a blended fluid field that preserves the divergence-free condition, hence an additional divergence-free projection follows the blending operation. One related method is presented by English et al. [2013], where a Voronoi partitioning is constructed to decide the extent of the valid refinement grids. Similarly, when multiple local grids cover the same global cell, we query the signed distance to each rigid body group with a level set constructed on the local grids to determine which local value to inherit from. We call this the *min-sdf* strategy, since it uses the minimum distance to the closest rigid body level set to assign flow field values.

4.4.2 Final Pressure Projection. The final projection step is the same as the previous approach: based on the geometry, we compute the divergence operator $\mathbf{D}_{h_G}\mathbf{L}_G$ on the global grid and $\mathbf{D_h}\mathbf{L}_{L_k}$ on the local grids; form the linear system in Eq. 1; solve it on the global grid G and project the fluid velocity by Eq. 2. Unlike the pressure solve on line 8 of Alg. 1, pressure field computed here (line 21 of Alg. 1) is immediately discarded after projection and not used in the subsequent local substepping.

5 RESULTS

In this section, we evaluate our proposed framework using several compelling examples and experiments. Unless otherwise stated, the global and local time step is $\Delta t_{\mathrm{G}} = 0.01$ s and $\Delta t = \frac{1}{4}\Delta t_{\mathrm{G}}$, and the grid size is 128^3 and 64^3 for the global and local grids, respectively. Each refinement grid contains a single rigid body and has an effective resolution of 256^3 compared to the global. We use 4 multigrid V-cycles on the coarse grid, and 30 red-black Gauss-Seidel iterations on the local subdomain for projection. All experiments were implemented in Python using Warp [Macklin 2022] and executed on an NVIDIA RTX 3060 GPU with 12 GB of memory.

Our rigid body simulation uses the extended position-based dynamics (XPBD) [Macklin et al. 2016] implementation available in Warp. Water surface is traced with narrow-band FLIP [Ferstl et al. 2016] with blending $\alpha = 0.99$. The level set of each rigid body ϕ_{B_k} is computed offline with OpenVDB. At run-time, the new level set on the global and local grids is generated by transforming the sample points to the rigid body frame and resampling ϕ_{B_k}.

5.1 Example Scenes

5.1.1 Boat Wake. The scene from the teaser Fig. 1, a propeller constantly stirs the water as a boat zooms through the water. Nearby buoys experience the effects of its wake. Rigid-fluid coupling generates buoyancy forces that keep the buoys afloat. The quantity of water being displaced by the boat motion creates a wave that later causes the buoys to move away from the wake. In the supplementary video, a comparison against a monolithic simulation with high resolution grid shows that qualitative simulation behavior is attained by our adaptive multigrid simulation scheme.

5.1.2 Fast Car. Two cars drive closely past each other, causing a stack of lightweight boxes to topple, as shown in Fig. 5. The scene is decomposed into multiple overlapping local grids and coupled through a coarse global grid encompassing the entire scene, and it demonstrates the ability to capture subtle behaviors achieved through rigid-fluid coupling.

5.1.3 Dam Break. An array of cubes begins free fall while the water collapses from the left. The water flows in and out of the local subdomains centered at each rigid body without being affected by the adaptive grids. This scene shows the ability of our method to handle many dynamic rigid objects in close proximity.

5.1.4 Flow Past Sphere. We compare reference simulations of a smoke plume interacting with a static sphere using a 64^3 coarse grid and 256^3 fine grid against our method using 64^3 coarse grid and 256^3 resolution adaptive region (see Fig. 4). We show that our method gets the same low-frequency motion as the fine reference, and adds more detail compared to the coarse reference. To isolate the influence of grid refinement, this example uses the same timestep for coarse and fine grids, i.e. $\Delta t_{\mathrm{G}} = \Delta t = 0.01\ s$.

5.2 Performance

The time consumption of a global step and a local step is recorded in Table 1. In all our demo scenes, our global simulator runs at an interactive rate (10 fps) and client-side update is real-time (<30 ms).

The residual of the multigrid solve is recorded in Table 2. With four V-cycles, our multigrid solver is able to reduce the divergence by three orders on average, and the local grid smoother achieves

Table 1: Computation time for each step of our pipeline. Timing results are reported in ms. The computation is divided into 4 stages: advection, computing the divergence operator D_h, computing and integrating the coupling pressure p_c, and divergence-free projection. Execution time for the global grid step and local grid step are recorded separately. As there are multiple local grids, the local time measures the worst-case execution time in all local grids. In the *Flow Past Sphere* scene, even under a synchronous update setting, the total time of our method (global + local + merge) is still more than 10 times lower than the fine grid reference.

		global					local					
scene	dimension	advect	D_h	p_c	project	total	advect	D_h	p_c	project	total	merge
Boat Wake	$128 \times 128 \times 64$	10.47	8.19	26.49	19.16	73.91	0.79	3.89	3.69	5.33	8.05	14.54
Fast Car	128^3	8.41	5.12	41.15	57.77	114.23	0.38	2.92	4.17	5.69	12.5	14.0
Dam Break	128^3	20.07	7.68	39.67	34.08	117.17	2.31	2.24	3.26	5.57	11.87	26.73
Flow Past Sphere, Fine	256^3	77.43	16.33	-	491.79	585.56	-	-	-	-	-	-
Flow Past Sphere	64^3	2.44	1.27	-	10.09	13.80	2.08	2.65	-	15.45	20.18	17.99

Table 2: Efficiency of our multigrid solver.

solver	dimension	avg. local error reduction	avg. global error reduction	convergence factor	V-cycle time (ms)
Boat Wake	$128 \times 128 \times 64$	0.0359	2.09×10^{-3}	0.214	2.68
Fast Car	128^3	0.0544	7.61×10^{-3}	0.295	6.96
Flow Past Sphere	64^3	0.0313	5.88×10^{-3}	0.277	3.01
Dam Break	128^3	0.0443	6.63×10^{-3}	0.285	2.21

Table 3: Divergence of each blending scheme for scene in Fig. 7. The lower the value, the closer to the desired incompressible divergence-free state. The proposed softmin-sdf scheme performs best, preserving the incompressibility of the fluid and a smooth transition between subdomain boundaries.

blending scheme	total divergence	boundary divergence
min-sdf ($\Delta t_G = 4\Delta t$)	2.43×10^{-6}	1.16×10^{-5}
min-sdf	1.34×10^{-5}	2.98×10^{-6}
softmin-sdf	2.36×10^{-5}	7.48×10^{-6}
linear blending	3.02×10^{-5}	1.50×10^{-5}
fine reference	5.84×10^{-8}	NA

two orders of divergence reduction, measured in the infinite norm. This gets us plausible fluid behavior.

5.3 Local-to-Global Fluid Blending

We progressively disable the asynchronous stepping and min-sdf blending in our local-to-global blending scheme in a simplified 2D simulation with three overlapping local grids. We record the L2-norm of the cell-centered divergence field before the final projection to quantify visual plausibility. We also record the divergence of the cells adjacent to the subdomain boundary to gauge the discrepancy at the boundary. A visual comparison is listed in Fig. 7. At a bare minimum, a naive linear blending is listed as a reference. The global grid is 256×256 with resolution $h_G = \frac{1}{256}$; each local grid is 256×256 with $h = \frac{1}{512}$. Lastly, we add a fine-resolution reference on a full 512×512 grid. Table 3 confirms the superiority of the min-sdf blending with frequency domain guiding in terms of both

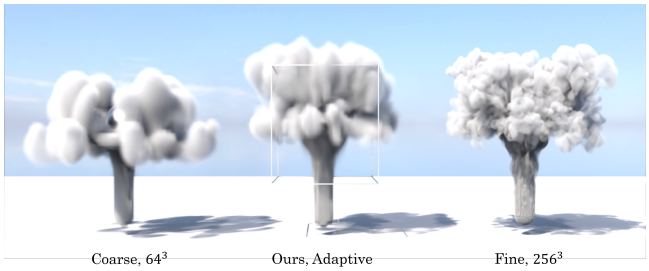


Figure 4: Flow Past Sphere scene. The results produced by our adaptive multigrid approach compared to global simulations using coarse (left plume) and fine (right plume) grids. Our method (middle plume) produces plausible high-frequency detail in adaptive regions, and there are no perceivable artifacts at boundaries or the global-local interface.

measurements. On the other hand, running a global simulation with $\Delta t_G = 4\Delta t$ doesn't deteriorate the visual plausibility.

6 CONCLUSION

We propose a framework for simulating fluids and rigid bodies designed for heterogeneous distributed computing architectures. Our approach relies on an adaptive multigrid scheme to minimize computational overhead on client devices, whereas a coarse grid spanning the entire scene efficiently manages global effects of the fluid. We believe that our framework paves the way for novel uses of distributed computing to power the next generation of interactive computer graphics applications by seamlessly leveraging the computing resources of a diverse array of devices for performing simulation.

ACKNOWLEDGMENTS

We thank Kyle Chand for his advice and many illuminating discussions about multigrid methods and fluid simulations.

REFERENCES

Marco Ament, Gunter Knittel, Daniel Weiskopf, and Wolfgang Strasser. 2010. A Parallel Preconditioned Conjugate Gradient Solver for the Poisson Problem on a Multi-GPU Platform. In *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*. 583–592. https://doi.org/10.1109/PDP.2010.51

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A Fast Variational Framework for Accurate Solid-Fluid Coupling. *ACM Trans. Graph.* 26, 3 (jul 2007), 100–es. https://doi.org/10.1145/1276377.1276502

Marsha J Berger and Phillip Colella. 1989. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics* 82, 1 (1989), 64–84.

Marsha J Berger and Joseph Oliger. 1984. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics* 53, 3 (1984), 484–512.

Alexander Brown, Gary Ushaw, and Graham Morgan. 2019. Aura projection for scalable real-time physics. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Montreal, Quebec, Canada) *(I3D '19)*. Association for Computing Machinery, New York, NY, USA, Article 1, 9 pages. https://doi.org/10.1145/3306131.3317021

Mark Carlson, Peter J. Mucha, and Greg Turk. 2004. Rigid Fluid: Animating the Interplay between Rigid Bodies and Fluid. *ACM Trans. Graph.* 23, 3 (aug 2004), 377–384. https://doi.org/10.1145/1015706.1015733

Nuttapong Chentanez and Matthias Müller. 2011. Real-Time Eulerian Water Simulation Using a Restricted Tall Cell Grid. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 82, 10 pages. https://doi.org/10.1145/1964921.1964977

Jieyu Chu, Nafees Bin Zafar, and Xubo Yang. 2017. A Schur Complement Preconditioner for Scalable Parallel Fluid Simulation. *ACM Trans. Graph.* 36, 4, Article 139a (jul 2017), 11 pages. https://doi.org/10.1145/3072959.3092818

Saulo Soares De Oliveira, Carlos Henrique R. Souza, Jefferson Carvalho Silva, and Sérgio T. Carvalho. 2024. Towards Scalable Cloud Gaming Systems: Decoupling Physics from the Game Engine. In *Proceedings of the 22nd Brazilian Symposium on Games and Digital Entertainment* (Rio Grande (RS), Brazil) *(SBGames '23)*. Association for Computing Machinery, New York, NY, USA, 151–160. https://doi.org/10.1145/3631085.3631225

R. Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013. Chimera grids for water simulation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Anaheim, California) *(SCA '13)*. Association for Computing Machinery, New York, NY, USA, 85–94. https://doi.org/10.1145/2485895.2485897

Ronald P. Fedkiw. 2002. Coupling an Eulerian Fluid Calculation to a Lagrangian Solid Calculation with the Ghost Fluid Method. *J. Comput. Phys.* 175, 1 (2002), 200–224. https://doi.org/10.1006/jcph.2001.6935

Florian Ferstl, Ryoichi Ando, Chris Wojtan, Rüdiger Westermann, and Nils Thuerey. 2016. Narrow band FLIP for liquid simulations. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics* (Lisbon, Portugal) *(EG '16)*. Eurographics Association, Goslar, DEU, 225–232.

Florian Ferstl, Rüdiger Westermann, and Christian Dick. 2014. Large-Scale Liquid Simulation on Adaptive Hexahedral Grids. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1405–1417. https://doi.org/10.1109/TVCG.2014.2307873

Nick Foster and Dimitri Metaxas. 1996. Realistic Animation of Liquids. *Graphical Models and Image Processing* 58, 5 (1996), 471–483. https://doi.org/10.1006/gmip.1996.0039

Frederic Gibou, Ronald P. Fedkiw, Li-Tien Cheng, and Myungjoo Kang. 2002. A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. *J. Comput. Phys.* 176, 1 (2002), 205–227. https://doi.org/10.1006/jcph.2001.6977

Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph.* 31, 6, Article 148 (nov 2012), 9 pages. https://doi.org/10.1145/2366145.2366167

Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. 2005. Coupling Water and Smoke to Thin Deformable and Rigid Shells. *ACM Trans. Graph.* 24, 3 (jul 2005), 973–981. https://doi.org/10.1145/1073204.1073299

Libo Huang, Ziyin Qu, Xun Tan, Xinxin Zhang, Dominik L. Michels, and Chenfanfu Jiang. 2021. Ships, splashes, and waves on a vast ocean. *ACM Trans. Graph.* 40, 6, Article 203 (dec 2021), 15 pages. https://doi.org/10.1145/3478513.3480495

Michael Lentine, Wen Zheng, and Ronald Fedkiw. 2010. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Trans. Graph.* 29, 4, Article 114 (jul 2010), 9 pages. https://doi.org/10.1145/1778765.1778851

Haixiang Liu, Nathan Mitchell, Mridul Aanjaneya, and Eftychios Sifakis. 2016. A Scalable Schur-Complement Fluids Solver for Heterogeneous Compute Platforms. *ACM Trans. Graph.* 35, 6, Article 201 (dec 2016), 12 pages. https://doi.org/10.1145/2980179.2982430

Miles Macklin. 2022. Warp: A High-performance Python Framework for GPU Simulation and Graphics. https://github.com/nvidia/warp. NVIDIA GPU Technology Conference (GTC).

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) *(MIG '16)*. Association for Computing Machinery, New York, NY, USA, 49–54. https://doi.org/10.1145/2994258.2994272

A. McAdams, E. Sifakis, and J. Teran. 2010. A Parallel Multigrid Poisson Solver for Fluids Simulation on Large Grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Madrid, Spain) *(SCA '10)*. Eurographics Association, Goslar, DEU, 65–74.

Charles S Peskin. 2002. The immersed boundary method. *Acta numerica* 11 (2002), 479–517. https://doi.org/10.1017/S0962492902000077

Doug Roble, Nafees bin Zafar, and Henrik Falt. 2005. Cartesian Grid Fluid Simulation with Irregular Boundary Voxels. In *ACM SIGGRAPH 2005 Sketches* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 138–es. https://doi.org/10.1145/1187112.1187279

Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128. https://doi.org/10.1145/311535.311548

Jos Stam. 2003. Real-Time Fluid Dynamics for Games. (05 2003).

Tsunemi Takahashi, Heihachi Ueki, Atsushi Kunimatsu, and Hiroko Fujii. 2002. The Simulation of Fluid-Rigid Body Interaction. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications* (San Antonio, Texas) *(SIGGRAPH '02)*. Association for Computing Machinery, New York, NY, USA, 266. https://doi.org/10.1145/1242073.1242279

Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. 2000. *Multigrid*. Elsevier.

Xinlei Wang, Yuxing Qiu, Stuart R. Slattery, Yu Fang, Minchen Li, Song-Chun Zhu, Yixin Zhu, Min Tang, Dinesh Manocha, and Chenfanfu Jiang. 2020. A Massively Parallel and Scalable Multi-GPU Material Point Method. *ACM Trans. Graph.* 39, 4, Article 30 (aug 2020), 15 pages. https://doi.org/10.1145/3386569.3392442

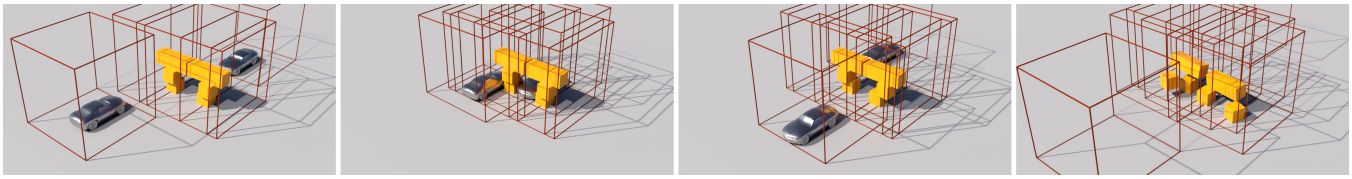


Figure 5: Fast Car scene. The disturbance caused by fast moving cars topples a stack of boxes. The scene is decomposed into multiple overlapping local grids and coupled through a coarse global grid encompassing the entire scene.

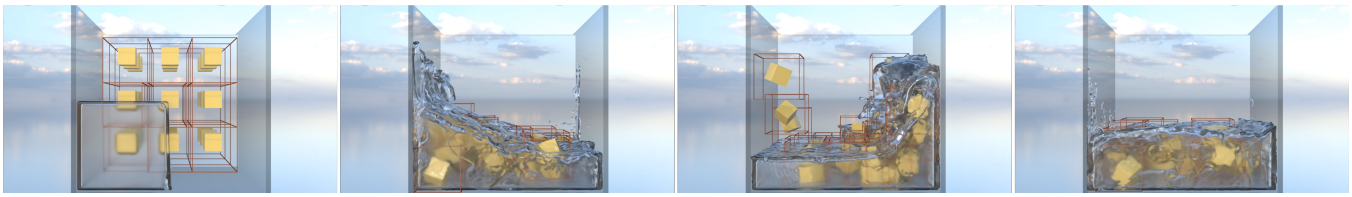


Figure 6: Dam Break Scene. An array of cubes begins free fall while the water collapses from the left. The water unaffectedly flows in and out of the local subdomains centered at each rigid body, while the rigid bodies and fluid exhibit plausible coupling.

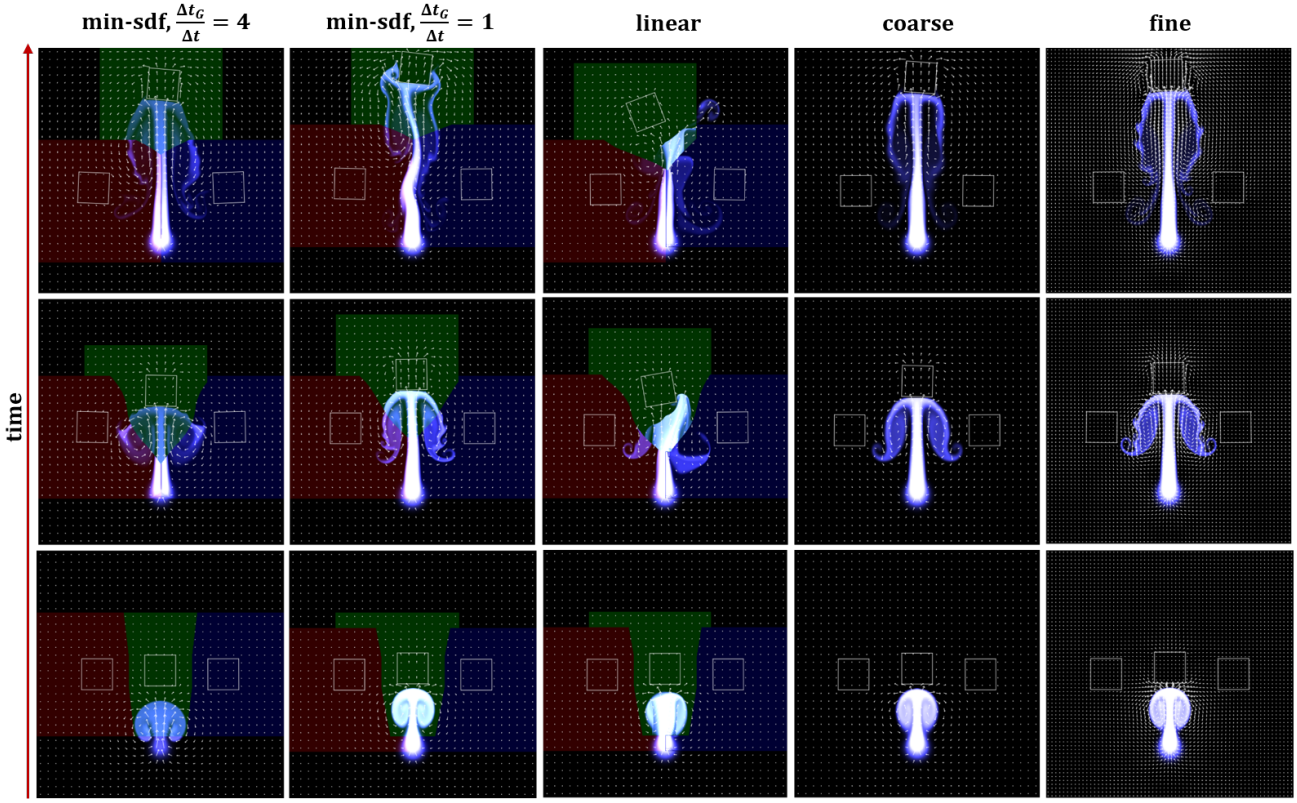


Figure 7: A steady stream flows against three floating cubes, which are simulated with three overlapping local grids. Different subdomains are highlighted in red, green and blue, respectively. From left to right: min-sdf, $\frac{\Delta t_G}{\Delta t} = 4$; min-sdf, $\frac{\Delta t_G}{\Delta t} = 1$; linear blending; 256×256 coarse grid reference; 512×512 unguided fine grid reference.