Fast Multi-Body Coupling for Underwater Interactions

T. Gao[1,2], X. Chen[1], X. Li[1], W. Li[3], B. Chen[1], Z. Pan[4], K. Wu[4] and M. Chu[1]

[1] Peking University, China
[2] Tsinghua University, China
[3] Shanghai Jiao Tong University, China
[4] LIGHTSPEED, USA

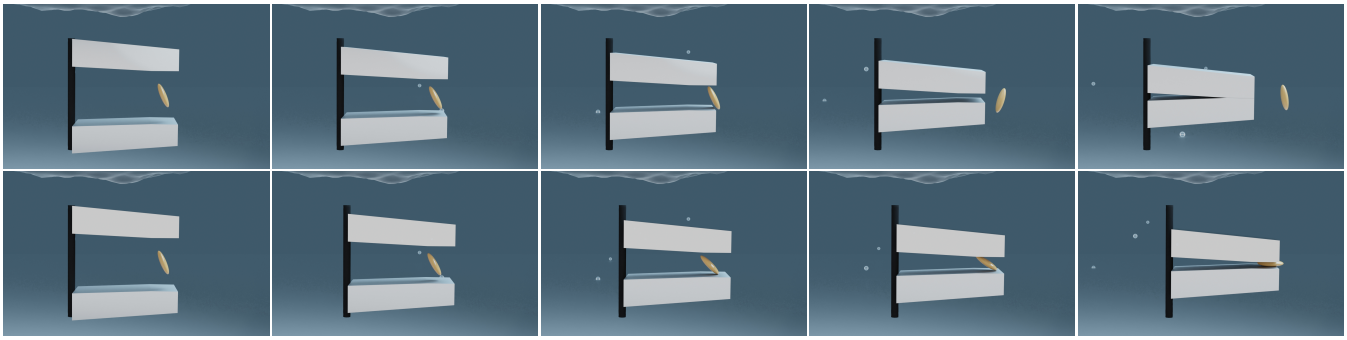


Figure 1: *An illustration of underwater grasping. Top: Naive control strategy designed for fluidless environments leads to a failed grasp, resulting in the fluid flow generated by the gripper pushes the target object away. Bottom: Fine-tuned trajectory achieves a successful grasp by taking fluid-mediated interactions into account.*

Abstract

Simulating multi-rigid-body interactions in underwater environments is crucial for various downstream applications, such as robotic navigation, manipulation, and locomotion. However, existing approaches either rely on computationally expensive volumetric fluid-rigid simulations or focus solely on single-body dynamics. In this work, we introduce a fast framework for simulating multi-rigid-body coupling in underwater environments by extending the added mass paradigm to capture global interactions in incompressible, irrotational fluids. Our method solves a Boundary Integral Equation (BIE) for the potential flow field, from which we derive the governing equation of motion for multiple underwater rigid bodies using a variational principle. We evaluate our method across a range of underwater tasks, including object gripping and swimming. Compared to state-of-the-art volumetric fluid solvers, our approach consistently reproduces similar behaviors while achieving up to 13× speedup. The example source code is available at https://github.com/guesss2022/fastMBCUI.

CCS Concepts

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

1. Introduction

Simulating underwater rigid body dynamics is critical for a wide range of applications, including physics-based animation [TGTL11], robotic locomotion and manipulation [LBH*22, XZX*23, WMS*23], and the modeling of bio-inspired underwater soft robots [MDZ*21, MWL*19]. These downstream tasks typically depend on sample-intensive algorithms such as model-predictive control and reinforcement learning, which place high demands on the efficiency of the underlying simulator. However, achieving both physical fidelity and computational efficiency for two-way coupled fluid and rigid bodies remains a significant challenge for existing simulation techniques.

On one hand, existing volumetric methods for fluid–rigid body two-way coupling rely on discretizing the fluid domain using grids [BBB07, LLDL21], particles [LXY*23], or hybrid Eulerian–Lagrangian representations [HFG*18]. However, accurately capturing fluid dynamics with these discretizations requires a high number of degrees of freedom, making them both memory-

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.
This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

intensive and computationally inefficient. In many cases, they involve solving large coupled systems for both fluid and rigid body velocities, with computational costs that scale superlinearly with the discretization resolution. On the other hand, to reduce computational costs, recent works [WP12, GSP*23] have proposed using added mass models to approximate the influence of the surrounding fluid on submerged rigid bodies through an additional generalized mass matrix. These methods significantly improve computational efficiency by avoiding explicitly solving for the volumetric fluid flow field. However, a major limitation is that the added mass matrix must be precomputed, which limits these methods to single rigid-body scenarios. As a result, fluid-mediated interactions between multiple rigid bodies are largely neglected, an omission that can lead to physically implausible behaviors or even failure in tasks such as robotic motion planning.

To achieve a balance between computational efficiency and physical accuracy, we introduce a fast framework for simulating multi-rigid-body coupling in underwater environments by extending the added mass paradigm to capture global interactions in incompressible, irrotational fluids. Unlike conventional added mass methods that rely on pose-invariant mass matrices and ignore inter-body coupling, our framework solves the incompressible, irrotational fluid velocity field at each timestep via a Boundary Integral Equation (BIE), enabling accurate multi-body interaction modeling. At the same time, our method inherits the key advantage of the added mass approach, avoiding expensive volumetric discretization of the fluid domain. Leveraging the computed potential field, we derive a coupled added mass model and formulate the resulting underwater dynamics using a variational approach grounded in Lagrangian mechanics. This leads to a physically consistent model that captures fluid-induced momentum exchange across multiple rigid bodies. To further accelerate the global solve in BIE, we adapt recent advances in variational preconditioning [CSD24] to our framework. The integration of this fast preconditioner significantly improves simulation performance in multi-body scenarios, even in the presence of complex geometries. Our framework bridges the gap between theory and practice for surface-only simulation of multi underwater rigid bodies, and our contributions can be summarized as follows:

- A numerical scheme for simulating multiple underwater rigid bodies using BIE.
- An accelerated differentiable BIE solver for underwater simulation using the variational preconditioner [CSD24].

We evaluate our framework across a variety of underwater scenarios involving multiple interacting rigid bodies. Compared to a baseline volumetric solver based on Lattice Boltzmann Method (LBM) [LLDL21], our method captures qualitatively similar multi-body fluid interaction effects that are entirely absent in prior single-body added mass models, while achieving a 13× speedup in runtime performance. We also demonstrate typical robotic scenarios, including swimming and grasping, where fluid-mediated interactions between rigid bodies critically influence the outcomes. These results highlight the effectiveness and efficiency of our approach.

2. Related Work

We review related techniques on the numerical solutions of partial differential equations for fluid dynamics.

Volumetric Discretization. Significant efforts have been made over the past decades to strike the optimal balance between efficacy and accuracy of fluid simulations. The prominent methods used in the graphics community discretize the volume of the fluid body in grid [BBB07, LLDL21], particle [ACAT13], or hybrid representation [ZB05, HFG*18]. Although these methods lead to full-featured simulators that capture all the details throughout the fluid body, they often come with a high computational cost that scales super-linearly with the resolution. Based on the volumetric discretization, a series of two-way solid-fluid coupling techniques has been developed. The two-way coupling technique for grid-based methods [BBB07] requires solving a coupled global linear system at each timestep to enforce incompressibility and solid-fluid boundary conditions. The coupling techniques for particle-based methods [AIA*12, ACAT13] exchange momentum between particles and solid so it does not involve linear system solvers. However, the timestep sizes for particle-based methods are typically much smaller, which again leads to a high computational cost. Recently, two-way coupling techniques for LBM [LLDL21] have been proven to capture fine details with a lower computational overhead. Even through, LBM still need to discretize the whole fluid simulation domain, leading to a much higher memory and computational overhead than ours.

Boundary Discretization. To avoid the high cost of volumetric discretization, various methods have been proposed to lower the cost by reducing from volumetric to boundary-based discretization. These techniques are based on the observation that, when the fluid is incompressible and irrotational, its velocity field corresponds to the solution of a Laplace equation, which can be found by solving a BIE. In [DHB*16], such assumption is taken to simulate free-surface flow using a surface-only discretization, but their method does not consider two-way solid-fluid coupling. Weißmann and Pinkall [WP12] show that the impact of irrotational fluid on a single rigid body can be precomputed as an added mass matrix. The formulation proposed in [KMRMH05] extends the added mass method to account for multiple rigid bodies, but did not analyze the computational efficiency of a practical implementation. In [GNS*12], the merits of boundary and volumetric discretization are combined. Compared with the aforementioned techniques, our focus is on the practical computational tractability and efficacy of a general, boundary-only solid-fluid coupling techniques. It is known [CSD24, DHB*16, SHW19, SBH22] that solving the BIE involves a dense linear system that couples all boundary elements, which is not necessarily faster than existing volumetric methods. Fortunately, the matrix-vector multiplication can be accelerated using algorithms such as the Fast Multipole Method (FMM) [YBZ04] or Hierarchical matrices (H-matrices) [BGH03], due to the special property of matrix entries. In addition, this dense linear system can be solved using iterative algorithms [SS86, Not00]. Recently, Chen et al. [CSD24] show that iterative algorithms can be significantly accelerated by using multilevel preconditioners.

Simplified Fluid Models. There exists other heuristic models that deliver plausible fluid animations, while being partially or even

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

non-physics-based. In [YHK07], for example, wave equations are used to model minor interruptions to a large open water space, and empirical force models are used for solid-fluid coupling. Follow-up works use the more accurate shallow water equations [CM10] and nonlinear wave equations [JSMF*18], but the solid-fluid coupling is again based on empirical force models. In [OKRC10], a force model is developed to animate underwater cloth motions, where the key idea is to use a so-called history-laden drag to model the history-dependent influences from the fluid body. It has been shown in various works [GSP*23, SPG*24, PM18] that drag force is a dominant part of solid-fluid interaction, and a heuristic local drag force model can deliver plausible solid-fluid coupling motions. While such a simple and efficient force model enables a range of downstream robotic applications, such as controller optimization for underwater locomotion [JLH*21, PM18, MWL*19], its inability to account for coupling between multiple rigid bodies significantly limits its applicability in more complex scenarios. Another important class of simplified fluid models is the panel method. Similar to our method, the panel method considers irrotational flow fields and uses boundary elements to approximate the flow potential. Variants of the panel method can also incorporate vortex panels or particles to satisfy circulation requirements [Eri90, BSC14, Car18, SSA*20, LSAM22]. However, these methods are primarily applied in the field of aerodynamic simulations, simulating the wake flow around aircraft wings, and focusing on the flow field rather than simulating the motion of solid objects within it. Therefore, while these methods offer higher accuracy for specific engineering scenarios, it is non-trivial to extend these methods to handle multiple coupled rigid bodies.

3. Simulation Formulation and Acceleration Framework

In this section, we propose a fast and physically consistent framework for simulating multiple rigid bodies interacting due to an incompressible, irrotational fluid in an unbounded domain. Such a fluid, where the velocity field equals the negative gradient of a scalar potential (hence called "potential flow"), represents the minimum total kinetic energy configuration among all divergence-free velocity fields satisfying the boundary conditions [WP12]. This assumption of minimum total kinetic energy, combined with velocity-matching boundary conditions, leads to a unique solution of the potential field and thus the velocity field.

In this section, we will begin by representing the fluid velocity field as a linear function of the generalized velocities of the rigid body system. We then formulate the kinetic energy of the potential flow as a function of the rigid-body system's coordinates and velocities. This allows the flow energy to be treated as an additional component of the kinetic energy of the rigid body system during simulation, allowing us to derive the equation of motion using Lagrangian mechanics. In practice, we solve for the fluid potential field using the method of fundamental solution by placing "point potential sources" inside the rigid body to approximate the external potential field. The strengths of these sources are then determined by solving the BIEs derived from velocity-matching boundary conditions. Unfortunately, in the multi-body setting, the fluid-added kinetic energy induces a mass-matrix that is changing over frames, unlike the constant matrix used by [WP12] when only a single rigid body is involved. Therefore, we need to solve the BIE in every frame, which becomes a computational bottleneck. At the end of this section, we will discuss our method to alleviate the computational burden by using the recently proposed variational preconditioner [CSD24].

3.1. Variational Formulation of Multi-Body Dynamics

We begin with a multi-rigid-body Lagrangian, which we later augment with fluid inertia. Each body i has a 6D configuration, i.e., the translation $\mathbf{T}_i \in \mathbb{R}^3$ and rotation $\mathbf{R}_i \in \mathrm{SO}(3)$. To simplify the formulation while avoiding singularities in SO(3), we adopt a local reparameterization strategy using a minimal 3D rotation vector $\theta_i \in \mathbb{R}^3$, such that the configuration becomes $\mathbf{q}_i = (\mathbf{T}_i, \theta_i)$ and its local rotation is described by the Rodrigues formula $\mathbf{R}_i(\theta_i) = \exp([\theta_i]_\times)\mathbf{R}_i^\star$, where $\mathbf{R}_i^\star$ denotes the nominal rotation about which the local chart is centered. The matrix $[\bullet]_\times$ is the skew-symmetric cross-product matrix associated with a vector. To ensure the validity of such parameterization, we always set the nominal rotation as the current state. This treatment is also known as differential rotation, and we refer readers to [RSCO25] for more details. With such reparameterization, the rigid body's kinematic configuration can be represented as a vector $\mathbf{q}$ and the velocity vector is represented as $\dot{\mathbf{q}}$, without requiring additional constraints. It is well-known that for such compact representation, the dynamic behavior of the system is uniquely determined by the Euler-Lagrange equation, applied on the following Lagrangian function [SD06]:

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{M}(\mathbf{q})\dot{\mathbf{q}} - V(\mathbf{q}), \tag{1}$$

where the first term models the kinetic energy, with $\mathbf{M}(\mathbf{q})$ being the generalized mass matrix, and the second term $V(\mathbf{q})$ is a user-defined potential energy. Applying the Euler-Lagrange equation then yields the following equation of motion:

$$\frac{1}{2}\frac{\partial \dot{\mathbf{q}}^T \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}}{\partial \mathbf{q}} - \frac{\partial V(\mathbf{q})}{\partial \mathbf{q}} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \frac{\partial \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}}{\partial \mathbf{q}}\dot{\mathbf{q}}. \tag{2}$$

Our local chart parameterization is valid because the system state change is small over a single timestep, so the state after a timestep stays within the local chart that parameterizes the rigid body state after one timetsep. Specifically, we first solve for $\ddot{\mathbf{q}}$ via:

$$\ddot{\mathbf{q}} \leftarrow \mathbf{M}(\mathbf{q})^{-1}\left[\frac{1}{2}\frac{\partial \dot{\mathbf{q}}^T \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}}{\partial \mathbf{q}} - \frac{\partial V(\mathbf{q})}{\partial \mathbf{q}} - \frac{\partial \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}}{\partial \mathbf{q}}\dot{\mathbf{q}}\right]. \tag{3}$$

We then time integrate the velocity and configuration as:

$$\dot{\mathbf{q}}_i \leftarrow \dot{\mathbf{q}}_i + \ddot{\mathbf{q}}_i \Delta t, \quad \mathbf{q}_i \leftarrow \mathbf{q}_i + \dot{\mathbf{q}}_i \Delta t, \quad \mathbf{R}_i \leftarrow \exp([\theta_i]_\times)\mathbf{R}_i^\star. \tag{4}$$

Finally, we reparameterize by setting $\mathbf{R}_i^\star \leftarrow \mathbf{R}_i$ and $\theta_i \leftarrow 0$. From the above update rules, we can immediately see that $\dot{\theta}_i$ is the discrete-time angular velocity of the ith rigid body. We can also derive the continuous-time angular velocity from our parameterization, using the results in [GY15] to derive:

$$\begin{aligned} \frac{d}{dt}\mathbf{R}_i(\theta_i) &= [\omega_i]_\times \mathbf{R}_i(\theta_i) \\ \omega_i &= \left[\frac{\mathbf{R}_i(\theta_i)\theta_i\theta_i^T + (\mathbf{I} - \mathbf{R}_i(\theta_i))[\theta_i]_\times}{\|\theta_i\|^2}\right]\dot{\theta}_i, \end{aligned} \tag{5}$$

with ω_i being the continuous-time angular velocity of the ith rigid body in the world frame. We further define the corresponding

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

velocity in the body-fixed reference frame using an overbar as $\dot{\overline{\mathbf{T}}}_i = \mathbf{R}_i(\theta_i)^T \dot{\mathbf{T}}_i$ and $\overline{\omega}_i = \mathbf{R}_i(\theta_i)^T \omega_i$.

3.2. Added Mass for a Single Underwater Rigid Body

In order to extend the rigid-body Lagrangian in Eq. 1 to include fluid inertia, we adopt the classical Kirchhoff's method [Lam24] to express the fluid's kinetic energy, originally an integral over the infinite domain, in terms of the rigid body's finite variables. In this way, we account for the influence of fluid with an "added mass" term $\mathbf{M}_f$, so that $\mathbf{M}(q) = \mathbf{M}_r + \mathbf{M}_f$ and the total Lagrangian still depend only on $\mathbf{q}$ and $\dot{\mathbf{q}}$.

Considering the single rigid body with index i, Eq. 2 reduces to the classical Newton-Euler equation, and we have the following kinetic energy of the rigid body $\mathcal{K}_r$ in the reference frame:

$$\mathcal{K}_r = \frac{1}{2}\begin{pmatrix} \dot{\overline{\mathbf{T}}}_i^T & \overline{\omega}_i^T \end{pmatrix} \mathbf{M}_{r,i} \begin{pmatrix} \dot{\overline{\mathbf{T}}}_i \\ \overline{\omega}_i \end{pmatrix} \quad \mathbf{M}_{r,i} = \begin{pmatrix} m_i \mathbf{I} & \\ & \mathbf{I}_i \end{pmatrix}. \tag{6}$$

Here I is the 3×3 identity matrix, m_i and $\mathbf{I}_i \in \mathbb{R}^{3\times 3}$ are the mass and inertia tensor in reference-frame, respectively, and $\mathbf{M}_{r,i} \in \mathbb{R}^{6\times 6}$ is the ith rigid body's generalized mass matrix. We follow Kirchhoff's method and consider the impact of an incompressible, irrotational fluid in the ambient space. By the Helmholtz-Hodge decomposition (see e.g. [STW24]), we can see that the velocity field of such fluid can be written as the negative gradient of a potential field ϕ, i.e., $-\nabla\phi$. Since there is only a single rigid body, we can solve for the potential fluid via the following Laplace equation in the reference frame:

$$\begin{cases} \phi(\mathbf{x}) = 0 & \|\mathbf{x}\| = \infty \\ -\overline{\mathbf{n}}_i(\mathbf{x})^T \nabla\phi(\mathbf{x}) = \overline{\mathbf{n}}_i(\mathbf{x})^T [\overline{\omega}_i \times \mathbf{x} + \dot{\overline{\mathbf{T}}}_i] & \mathbf{x} \in \partial\overline{\Omega}_i \\ \Delta\phi(x) = 0 & \mathbf{x} \in -\overline{\Omega}_i \end{cases}, \tag{7}$$

where $\Omega_i \subset \mathbb{R}^3$ is the space taken by the ith rigid body, $-\Omega_i$ is the complement, and $\mathbf{n}_i(\mathbf{x})$ is the outward normal of the ith rigid body at $\mathbf{x}$, all in world space. The first and second equations define the boundary condition of the Laplace equation, where the first equation requires the potential to vanish at infinity, while the second equation requires the fluid velocity to match the rigid body velocity at the common boundary. Similarly to velocity variables, we can define the corresponding variable in the reference frame as $\overline{\Omega}_i = \mathbf{R}_i^T(\Omega_i - \mathbf{T}_i)$ and $\overline{\mathbf{n}}_i(\mathbf{x}) = \mathbf{R}_i^T \mathbf{n}_i(\mathbf{R}_i\mathbf{x} + \mathbf{T}_i)$. In our Laplace equation, the first equation is the boundary condition at infinity, the second equation requires the fluid velocity to match that of the ith rigid body, and the third equation requires the velocity field to be incompressible. We can then derive the corresponding kinetic energy of the fluid body $\mathcal{K}_f$ by the following volume integral:

$$\begin{aligned} \mathcal{K}_f(\mathbf{q},\dot{\mathbf{q}}) &= \frac{\rho}{2}\int_{-\overline{\Omega}_i} \|\nabla\phi\|^2 d\mathbf{x} \\ &= \frac{\rho}{2}\int_{-\overline{\Omega}_i} \nabla\cdot(\phi\nabla\phi) d\mathbf{x} \\ &= -\frac{\rho}{2}\int_{\partial\overline{\Omega}_i} \phi\nabla\phi\cdot d\mathbf{s} \\ &= \frac{\rho}{2}\int_{\partial\overline{\Omega}_i} \phi[\overline{\omega}_i \times \mathbf{x} + \dot{\overline{\mathbf{T}}}_i]\cdot d\mathbf{s}. \end{aligned} \tag{8}$$

In the second equality above, we use the following identity $\nabla\cdot(\phi\nabla\phi) = \nabla\phi\cdot\nabla\phi + \phi\Delta\phi = \|\nabla\phi\|^2$, where we use our third condition in Eq. 7 to yield $\Delta\phi = 0$. The third equality above is derived using the divergence theorem. Finally, we use our boundary condition of ϕ to derive the fourth equality. Since ϕ is linear in the rigid body's velocity $\overline{\mathbf{T}}_i$ and $\overline{\omega}_i$ in its reference frame, we can see that $\mathcal{K}_f$ is a quadratic form in the velocity as well, leading to the succinct formula:

$$\mathcal{K}_f(\mathbf{q},\dot{\mathbf{q}}) = \frac{1}{2}\begin{pmatrix} \dot{\overline{\mathbf{T}}}_i^T & \overline{\omega}_i^T \end{pmatrix} \mathbf{M}_{f,i} \begin{pmatrix} \dot{\overline{\mathbf{T}}}_i \\ \overline{\omega}_i \end{pmatrix}, \tag{9}$$

where $\mathbf{M}_{f,i}$ is the fluid's added mass matrix to the ith rigid body. Our system is closed by combining the two kinetic terms in Eq. 6 and Eq. 8:

$$\frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}(\mathbf{q})\dot{\mathbf{q}} = \mathcal{K}_r(\mathbf{q},\dot{\mathbf{q}}) + \mathcal{K}_f(\mathbf{q},\dot{\mathbf{q}}). \tag{10}$$

For a single underwater rigid body, we note that Eq. 7 endows the same problem data in the reference frame of the rigid body, thus $\mathbf{M}_{f,i}$ can be made invariant to the pose $\mathbf{q}$ in the reference frame, allowing it to be precomputed via a proper discretization of Eq. 7 and thereby enabling efficient simulation [WP12].

3.3. Fluid-Mediated Multi-Body Coupling

Our next goal is to account for more than one rigid body by deriving the associated kinetic energy. Without considering the fluid, the kinetic energy is the accumulation of all the rigid bodies:

$$\mathcal{K}_r(\mathbf{q},\dot{\mathbf{q}}) = \frac{1}{2}\sum_i \begin{pmatrix} \dot{\overline{\mathbf{T}}}_i^T & \overline{\omega}_i^T \end{pmatrix} \mathbf{M}_{r,i} \begin{pmatrix} \dot{\overline{\mathbf{T}}}_i \\ \overline{\omega}_i \end{pmatrix}. \tag{11}$$

For computational efficacy, authors of [WP12] proposed to accumulate the reference-frame added mass term and define the coupled kinetic energy as:

$$\frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}(\mathbf{q})\dot{\mathbf{q}} = \frac{1}{2}\sum_i \begin{pmatrix} \dot{\overline{\mathbf{T}}}_i^T & \overline{\omega}_i^T \end{pmatrix} (\mathbf{M}_{r,i} + \mathbf{M}_{f,i}) \begin{pmatrix} \dot{\overline{\mathbf{T}}}_i \\ \overline{\omega}_i \end{pmatrix}. \tag{12}$$

However, although the above computation is efficient, it ignores the fluid-mediated interactions between different rigid bodies, which could play a significant role for large bodies or bodies in close proximity. Indeed, when more than one rigid bodies exist, the boundary condition in Eq. 7 takes a different form at every timestep. We thus have to solve for the potential field in the world frame every time, via the following Laplace equation:

$$\begin{cases} \phi(\mathbf{x}) = 0 & \|\mathbf{x}\| = \infty \\ -\mathbf{n}_i(\mathbf{x},\mathbf{q})^T \nabla\phi(\mathbf{x}) = \mathbf{n}_i(\mathbf{x},\mathbf{q})^T [\omega_i \times \mathbf{x} + \dot{\mathbf{T}}_i] & \mathbf{x} \in \partial\Omega_i(\mathbf{q})\ \forall i \\ \Delta\phi(x) = 0 & \mathbf{x} \in -\cup_i \Omega_i(\mathbf{q}) \end{cases}, \tag{13}$$

where the key difference lies in the use of ω_i and $\mathbf{T}_i$ in the world frame instead of $\overline{\omega}_i$ and $\overline{\mathbf{T}}_i$ in the reference frame. Applying the divergence theorem, and we derive the expression for the fluid's kinetic energy as in Eq. 8 but in the world frame:

$$\mathcal{K}_f(\mathbf{q},\dot{\mathbf{q}}) = \frac{\rho}{2}\sum_i \int_{\partial\Omega_i} \phi[\omega_i \times \mathbf{x} + \dot{\mathbf{T}}_i]\cdot d\mathbf{s}. \tag{14}$$

Again, we see that $\mathcal{K}_f$ is a quadratic form in the concatenated velocity over all rigid bodies, which could be written as:

$$\mathcal{K}_f(\mathbf{q},\dot{\mathbf{q}}) = \frac{1}{2}\begin{pmatrix} \dot{\mathbf{T}}^T & \omega^T \end{pmatrix} \mathbf{M}_f(\mathbf{q}) \begin{pmatrix} \dot{\mathbf{T}} \\ \omega \end{pmatrix}, \tag{15}$$

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

where $\mathbf{M}_f(\mathbf{q})$ is the pose-dependent fluid-added mass matrix that strongly couples the velocities over all the rigid bodies, with $\mathbf{T}$ and ω being the concatenated world-space velocity. As compared with Eq. 9, Eq. 15 couples all the rigid bodies together, where $\mathbf{M}_f$ is a dense matrix with all non-zero entries in the general case. In particular, the off-diagonal blocks correspond to the velocity-level coupling between two different rigid bodies. The presence and variations of these terms results in momentum transfer between the bodies, thereby approximating their hydrodynamic interactions.

3.4. Boundary Discretization

In this section, we propose an efficient scheme to compute the potential field and the associated fluid-added mass matrix by solving a BIE in the least squares sense. Specifically, we represent $\partial\Omega_i$ using a triangle mesh with $\mathbf{x}_{ij}$ being the world-space center of the jth face of the ith rigid body. We follow [WP12, CSD24] and discretize ϕ using the Method of Fundamental Solution (MFS). Specifically, we introduce the following 3D fundamental solution of Eq. 13:

$$\Phi(\mathbf{x},\ \mathbf{y}) = \frac{1}{\|\mathbf{x}-\mathbf{y}\|}, \tag{16}$$

centered at some source point $\mathbf{y}$, which automatically satisfies the Laplace equation, except for the singularity $\mathbf{x}=\mathbf{y}$, and the boundary condition at infinity. To avoid the singularity and satisfy the boundary condition at the surface of all rigid bodies, we define the potential field by the following linear combination:

$$\phi(\mathbf{x}) = \sum_i \sum_j \alpha_{ij} \Phi(\mathbf{x},\ \mathbf{x}_{ij} - \mathbf{n}_i(\mathbf{x}_{ij})\epsilon). \tag{17}$$

Specifically, we introduce one source term for each triangular face of the rigid body surface, which is weighted by the unknown coefficient α_{ij}, and we assume the final potential field is defined by the accumulation of all the weighted potential source terms. Unfortunately, it is well-known that the function Φ has a singularity at $\mathbf{x}=\mathbf{y}$ where $\Phi=\infty$. To avoid singularity, we slightly move the source point within the rigid body along the normal direction $\mathbf{n}_i(\mathbf{x}_{ij})$ by a small offset ϵ. To nail down the exact potential field value, we only need to determine the unknown weights α_{ij}. We can solve for all α_{ij} by requiring the boundary condition in Eq. 13 to be satisfied at the center of each rigid body's surface triangle. This can be done by requiring the second line of Eq. 13 to be satisfied at every surface triangle center, denoted as x_{kl}. By plugging $x = x_{kl}$, we derive the following dense linear system, which can be solved for all α_{ij}:

$$\begin{pmatrix} \vdots \\ \cdots\ \mathbf{n}_k(\mathbf{x}_{kl}) \cdot \nabla_{\mathbf{x}} \Phi(\mathbf{x}_{kl}, \mathbf{x}_{ij} - \mathbf{n}_i(\mathbf{x}_{ij})\epsilon)\ \cdots \\ \vdots \end{pmatrix} \begin{pmatrix} \vdots \\ \alpha_{ij} \\ \vdots \end{pmatrix} =$$

$$\mathbf{K}_f^{\nabla}(\mathbf{q})\alpha = \mathbf{b}(\mathbf{q}) = \begin{pmatrix} \vdots \\ \mathbf{n}_k(\mathbf{x}_{kl})[\omega_k \times \mathbf{x}_{kl} + \dot{\mathbf{T}}_{kl}] \\ \vdots \end{pmatrix}. \tag{18}$$

Plugging in the solution α into the kinetic energy $\mathcal{K}_f$ in Eq. 15, we derive the discrete fluid kinetic energy as follows:

$$\mathcal{K}_f(\mathbf{q}, \dot{\mathbf{q}}) = \frac{\rho}{2} \mathbf{b}(\mathbf{q})^T \mathbf{D} \mathbf{K}_f(\mathbf{q}) \left[\mathbf{K}_f^{\nabla}(\mathbf{q})\right]^{-1} \mathbf{b}(\mathbf{q}). \tag{19}$$

where the matrix $\mathbf{K}_f$ is defined similarly to $\mathbf{K}_f^{\nabla}$, with the kl, ijth entry being $\Phi(\mathbf{x}_{kl}, \mathbf{x}_{ij} - \mathbf{n}_i(\mathbf{x}_{ij})\epsilon)$ and $\mathbf{D}$ is the diagonal matrix with the ijth diagonal entry being the jth face area on the ith rigid body. Since Eq. 19 is again a quadratic form in $\dot{\mathbf{q}}$, equation 11 still applies, and thus we have completed the derivation of our space-time discretized governing equation by combining Eqs. 3, 4, 5, 11, and 19. A brute-force algorithm for computing matrix-vector products with $\mathbf{K}_f$, $\mathbf{K}_f^{\nabla}$, and solving the linear system can be quite costly as the size of these matrices scales with the number of triangular faces on the discrete boundary of all rigid bodies. In the next section, we discuss convenient and efficient implementations of these operators.

3.5. Efficient Computation Scheme

It can be quite involved to derive all the terms required to evaluate Eq. 3 by hand. More importantly, the brute-force computation of matrices $\mathbf{K}_f$, $\mathbf{K}_f^{\nabla}$, and their inverse can be quite costly, since they are dense matrices with a size proportional to the number of triangles for discretizing the rigid body surfaces. To mitigate such a high cost and avoid the involved derivation, we propose to use the automatic differentiation tool [KMKM21], with a customized operator for the matrix-vector product with the matrix $[\mathbf{K}_f^{\nabla}]^{-1}$ that scales with the number of triangles. For evaluating $[\mathbf{K}_f^{\nabla}]^{-1}\mathbf{b}$, we adopt the iterative method to solve the dense linear system. Since the left-hand side is not symmetric positive definite in general, we need to use the more costly iterative solvers such as GMRES [SS86]. However, our experiments show that it is practically faster to form a linear square linear system and solve for:

$$\left[[\mathbf{K}_f^{\nabla}]^T [\mathbf{K}_f^{\nabla}]\right]^{-1} [\mathbf{K}_f^{\nabla}]^T \mathbf{b}, \tag{20}$$

for which we could use the conjugate gradient solver with faster iterations. To further accelerate the solver, we adopt the multi-level preconditioner [CSD24], which forms a hierarchy of sparse linear systems, each of which involve a series of small dense systems that can be solved in parallel. Finally, we note that Eq. 3 requires the evaluation of:

$$\frac{\partial [\mathbf{K}_f^{\nabla}(\mathbf{q})]^{-1}\mathbf{b}}{\partial \mathbf{q}} = -[\mathbf{K}_f^{\nabla}(\mathbf{q})]^{-1} \frac{\partial \mathbf{K}_f^{\nabla}(\mathbf{q})}{\partial \mathbf{q}} [\mathbf{K}_f^{\nabla}(\mathbf{q})]^{-1}\mathbf{b}, \tag{21}$$

for which we can implement a faster scheme as well. Specifically, the multiplication with $[\mathbf{K}_f^{\nabla}(\mathbf{q})]^{-1}$ can be evaluated using PCG.

In summary, the workflow of our algorithm is illustrated in Alg. 1 and Fig. 2, where matrix $\mathbf{F}$ builds $\mathbf{b}$ from general velocity $\mathbf{v} = \dot{\mathbf{q}}$ by $\mathbf{b} = \mathbf{F}\mathbf{v}$, and $\mathbf{A} = [\mathbf{K}_f^{\nabla}(\mathbf{q})]^{-1}\mathbf{F}$. The time derivative of $\mathbf{M}_f$ is computed through automatic differentiation, which offers better accuracy despite more computation overhead.

4. Results

All experiments are conducted on a system equipped with an 11th Gen Intel Core i7-11800H CPU, 16GB RAM, and an NVIDIA GeForce RTX 3060 GPU (6GB VRAM). Our method is implemented in PyTorch, leveraging its automatic differentiation capabilities to compute differential terms in dynamic equations. To accelerate linear solves, we develop custom Python operators in C++ and CUDA. We adopt Pinocchio [CSB*19] for articulated body

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

Algorithm 1: Fast Multi-Body Underwater Simulation

Input : Initial configuration $\mathbf{q}_0$, initial velocity $\mathbf{v}_0$
Time step Δt, total simulation time T
Fluid density ρ, rigid body meshes $\{\overline{\mathbf{V}}^i, \overline{\mathbf{F}}^i\}_{i=1}^N$
Output: Trajectories $\mathbf{q}(t)$, $\mathbf{v}(t)$ for $t \in [0,T]$
Load rigid body models;
$\mathbf{q} \leftarrow \mathbf{q}_0$, $\mathbf{v} \leftarrow \mathbf{v}_0$;
for *each rigid body* $i = 1$ *to* N **do**
  $\overline{\mathbf{n}}^i \leftarrow$ compute_normal$(\overline{\mathbf{V}}^i, \overline{\mathbf{F}}^i)$;
  $\overline{\mathbf{S}}^i = \overline{\mathbf{V}}^i - \epsilon \overline{\mathbf{n}}^i$; // Source points
end
$\mathbf{D} \leftarrow$ compute_area$(\{\overline{\mathbf{V}}^i\}_{i=1}^N, \overline{\mathbf{F}}^i\})$;
for $k = 0$ *to* $T/\Delta t$ **do**
  for *each rigid body* $i = 1$ *to* N **do**
    $\mathbf{R}^i, \mathbf{t}^i, \mathbf{Q}_p \leftarrow$ forward_kinematics$(\mathbf{q})$;
    // $\mathbf{Q}_p$ is the kinematic Jacobian
    $\mathbf{V}^i \leftarrow \mathbf{R}^i \cdot \overline{\mathbf{V}}^i + \mathbf{t}^i$;
    $\mathbf{S}^i \leftarrow \mathbf{R}^i \cdot \overline{\mathbf{S}}^i + \mathbf{t}^i$;
    $\mathbf{n}^i \leftarrow \mathbf{R}^i \cdot \overline{\mathbf{n}}^i$;
    $\mathbf{C}^i \leftarrow$ compute_face_centers$(\mathbf{V}^i, \overline{\mathbf{F}}^i)$;
    $\mathbf{C}_r^i \leftarrow \mathbf{C}^i - \mathbf{t}^i$;
  end
  $\mathbf{F} \leftarrow$ RHS_by_kinematics$(\{\mathbf{C}_r^i\}, \{\mathbf{n}^i\}, \mathbf{Q}_p)$;
  $\mathbf{K}_f^{\nabla} \leftarrow$ single_layer_gradient$(\{\mathbf{C}^i\}, \{\mathbf{S}^i\}, \{\mathbf{n}^i\})$;
  // gradient of single layer potential matrix in Eq.18
  $\mathbf{A} \leftarrow$ MinvF$(\mathbf{K}_f^{\nabla}, \mathbf{F})$; // use custom operator
  $\mathbf{K}_f \leftarrow$ single_layer_potential$(\{\mathbf{C}^i\}, \{\mathbf{S}^i\})$;
  $\mathcal{K}_f, \mathbf{M}_f \leftarrow$ assemble_fluid_energy$(\mathbf{D}, \mathbf{F}, \mathbf{A}, \mathbf{K}_f, \rho)$;
  // Eq.19
  $\mathbf{M}_r \leftarrow$ compute_rigid_mass_matrix$(\mathbf{q})$;
  $\mathbf{f}_b \leftarrow$ rnea$(\mathbf{q}, \mathbf{v}, \mathbf{0})$; // Rigid body forces by Recursive Newton-Euler Algorithm
  $\dot{\mathbf{M}}_f \leftarrow$ compute changing rate of $\mathbf{M}_f$;
  $\mathbf{f}_{bf} \leftarrow \nabla_{\mathbf{q}} \mathcal{K}_f - \dot{\mathbf{M}}_f \mathbf{v}$; // Fluid-induced forces
  $\tau_{\text{control}} \leftarrow$ compute_control_inputs$(\mathbf{q}, \mathbf{v})$; // Query robot controller for joint torques
  $\mathbf{a} \leftarrow (\mathbf{M}_r + \mathbf{M}_f)^{-1}(\mathbf{f}_{bf} - \mathbf{f}_b + \tau_{\text{control}})$; // Eq.3
  $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{a}\Delta t$;
  $\mathbf{q} \leftarrow$ integrate$(\mathbf{q}, \mathbf{v}\Delta t)$; // Eq.4
end

computations and use PyBullet for contact handling [CB21]. The timestep size is set to 0.01s for all experiments. We provide all relevant statistics in table 1.

Comparison with [LWP*23]. We first compare our method against a state-of-the-art Home-LBM [LWP*23], which requires discretizing the entire simulation domain of volumetric fluid. To ensure a fair comparison, we augment our model with an approximated viscous effect by introducing localized linear drag forces, following the approach of [GSP*23]. The drag exerted on an area

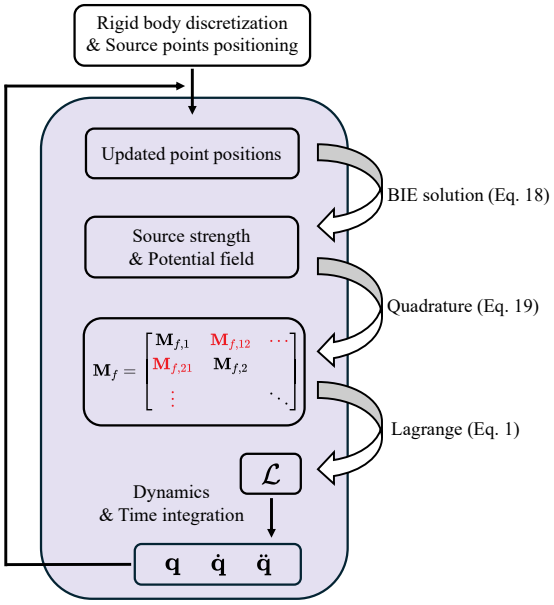


Figure 2: *Our workflow. The red parts in $\mathbf{M}_f$ are the key non-zero blocks that cause coupling. The shaded part represents the simulation loop. The operations in the rightmost column are all differentiable, enabling the use of derivatives of the Lagrange function with respect to coordinates in the dynamics.*

of the object's surface ($\Delta \mathbf{f}_d$) is approximated to be proportional to the component of the local rigid body velocity $\mathbf{v}$ along the surface, scaled by fluid density ρ and control area ΔA, as shown in Eq. 22, where the coefficient ν can be used to adjust the viscosity of the liquid (Fig. 4). The comparison is performed on a scenario where a heavy sphere falls onto a plane and generates fluid motion that influences an oblate spheroidal coin.

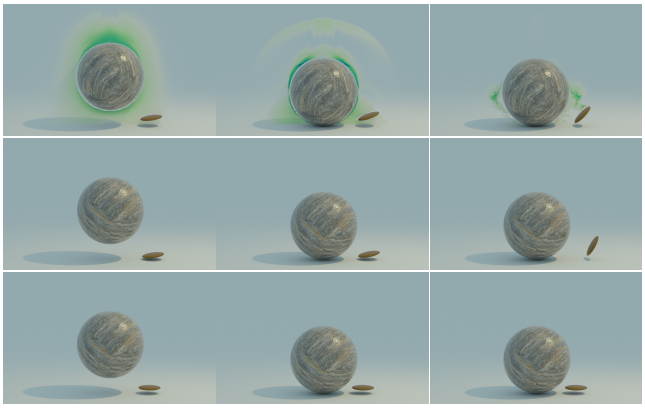
$$\Delta \mathbf{f}_d = -\nu\rho[\mathbf{v} - (\mathbf{v}\cdot\mathbf{n})\mathbf{n}]\Delta A \tag{22}$$

As shown in Fig. 3, our method produces results that are comparable to LBM in terms of both rigid body trajectories and physical details. Specifically, as the heavy sphere approaches the ground, the displaced fluid generates an upward flow that induces a counterclockwise rotation in the coin. We also visualize the fluid velocity magnitude field to reveal fine-scale interactions between the flow field and rigid bodies in the LBM. Our method reproduces similar rigid body dynamics without requiring volumetric simulation.

In terms of performance, our method takes 0.31 seconds per frame. In contrast, LBM requires a $225 \times 405 \times 450$ grid to achieve similar fidelity, resulting in over 4 seconds per frame. In terms of memory cost, our method requires 2088MB of GPU memory, while LBM takes 4298MB. Under a lower resolution, LBM can no longer resolve the fine-grained flow details (e.g., the coin-ground gap), resulting in artifacts such as the coin stuck to the ground (Fig. 5). While the LBM baseline uses a highly optimized GPU implementation, our method, implemented in Python with custom C++/CUDA kernels, still achieves an order-of-magnitude speedup without compromising result quality. We also compared with the performance of [SPG*24]. Although their approach requires only a few millisec-

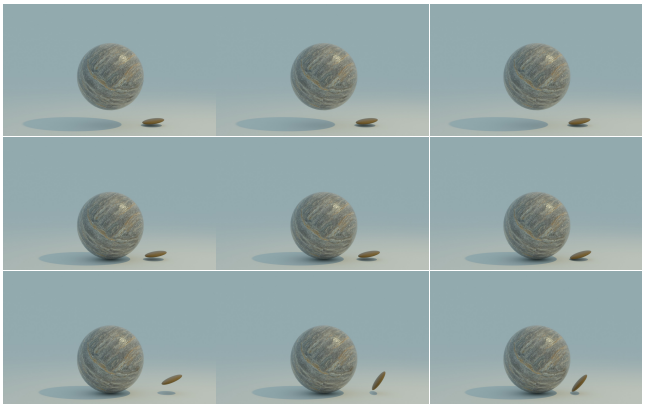
© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

onds of simulation time per frame, it completely fails to achieve coupling effects. Employing [CSD24]'s preconditioner in this case achieves 40% overall speedup over PCG with the Jacobi preconditioner. This is because the former leads to faster convergence (Fig. 6), requiring significantly fewer PCG iterations to reach certain threshold, despite the additional time needed to construct the preconditioner.



Frame 137 Frame 166 Frame 234

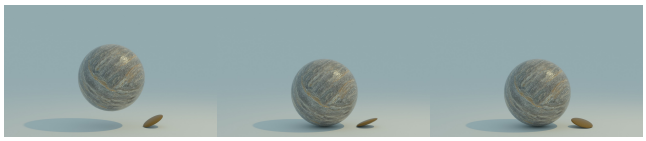
Figure 3: *Falling ball. Top, as the heavy sphere approaches the ground, the displaced fluid generates an upward flow via LBM that induces a counterclockwise rotation in the coin. We also visualize the magnitude of the velocity field for the LBM output, where green indicates a larger magnitude. Middle, without performing volumetric fluid simulation, our method produces a comparable result, while being 13× faster. Bottom, method in [SPG*24] fails to capture coupling effects due to low grid resolution.*



$\nu = 6 \times 10^{-3}$ $\nu = 1.2 \times 10^{-2}$ $\nu = 2.4 \times 10^{-2}$

Figure 4: *Falling ball with different viscosity. Unit of* ν *is* m/s. *The increased viscosity causes the coin to stabilize more quickly, as evidenced by the distance it is pushed away. Top, frame 137. Middle, frame 156. Bottom, frame 239.*

Comparison with [BBB07]. We further compare our method with a variational fluid-solid coupling framework [BBB07] using a simple experiment in which a cube moves through a fluid with an initial upward velocity. Both our method and the LBM allow the cube to move along the initial velocity direction over a long distance. In contrast, the cube with the method from [BBB07] stops earlier due to large energy dissipation from grid-based discretization and NS solver (Fig. 7). This highlights the limitations of that approach when simulating fully submerged rigid bodies. The result of [SPG*24] also exhibits significant dissipation due to the use of an empirical drag model. In terms of performance, [BBB07] requires over 4 seconds per frame using a $75 \times 300 \times 75$ grid to produce a stable result, [LWP*23] needs 0.22 seconds per frame with a $100 \times 400 \times 100$ grid, whereas added-mass-based methods require about 1ms per frame, as the added-mass tensors are constant in single-body cases and can be precomputed.



Frame 137 Frame 155 Frame 249

Figure 5: *Falling ball with lower resolution LBM. The coin inaccurately adheres to the ground during the sphere's descent. This experiment used a resolution of 200 × 360 × 400, with a runtime of 2.6 seconds per frame.*

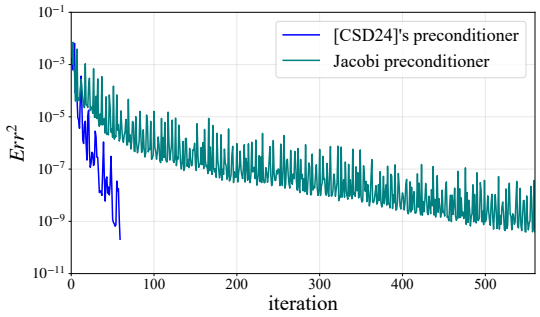
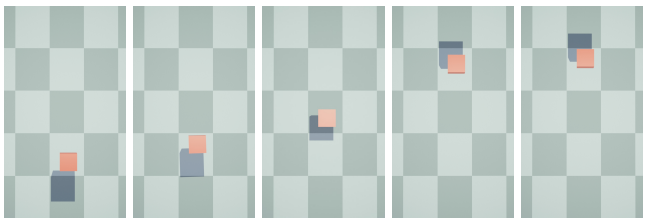


Figure 6: *PCG convergence behavior comparison. The blue line reaches the threshold and terminates after 59 iterations.*



Init [BBB07] [SPG*24] [LWP*23] Ours

Figure 7: *Cube pushing. Given a simple test that the cube is initialized with an upward velocity, both our method and LBM [LWP*23] move the cube along the initial direction with a long distance before stopping, but the cube with the method in [BBB07] or [SPG*24] can only move a small distance due to large energy dissipation.*

Underwater grasping. Gripper manipulation is a fundamental

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

challenge in robotic control, and its complexity increases significantly in fluid environments. In particular, gripper motion perturbs the surrounding fluid, which in turn affects the dynamics of the target object, as illustrated in Fig. 1 top. As a result, naive control strategies designed for dry environments often fail underwater; for instance, the closing motion of a gripper can generate fluid flow that pushes the target object away, leading to a failed grasp. However, by manually fine-tuning the grasp trajectory, e.g., moving the gripper slowly in the same direction as the coin as shown in Fig. 1 bottom, it is possible to achieve a successful grasp. This example highlights a promising application of our framework: enabling underwater robotic manipulation tasks for training deep learning algorithms in physically realistic environments.

Swimming. Using the controller described in [JLH*21], we can simulate the scenario where an articulated rigid-body robotic fish starts swimming from rest in a potential flow environment. The system consists of 3,840 triangular mesh elements, with each frame of simulation taking 0.4 seconds. Building on this, we can simulate the mutual influence between the robotic fish and other underwater objects through fluid interactions. First, a small fish is modeled as a passive rigid body, allowing us to observe how the robot's swimming flow affects nearby bodies. Fig. 8 illustrates such an interaction, where the passive fish is carried by the surrounding flow and effectively pushed away from the robot, avoiding collision without any active control. Next, in a more dynamic scenario, the robotic fish swims adjacent to a larger fish executing oscillatory motions, as shown in Fig. 9. The vigorous motion of the latter significantly disrupts the robot's trajectory, causing it to drift laterally. These examples demonstrate how fluid-mediated interactions can both affect and be affected by the robot's motion. Accurately capturing these complex dynamics is therefore essential for informing the design and training of underwater control strategies.

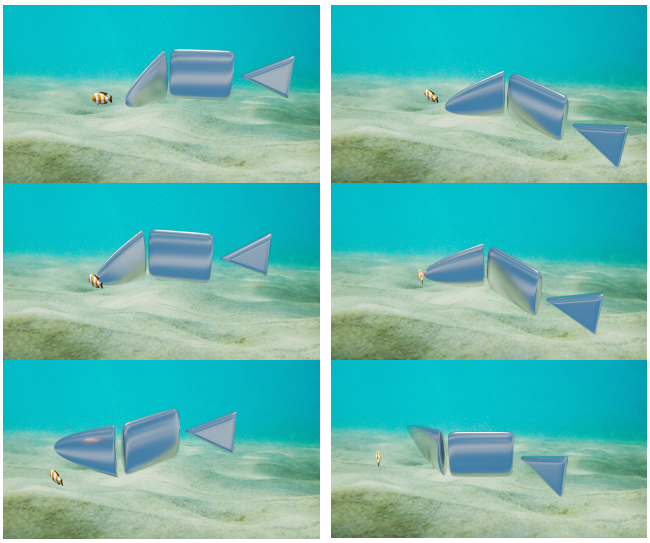


Figure 8: *The impact of the swimming robotic fish on the small fish's movement from two perspectives. Note that no collision occurred in this example.*

Scalability test. To evaluate the efficiency of our accelerated algorithm, we benchmark its scalability against native PyTorch

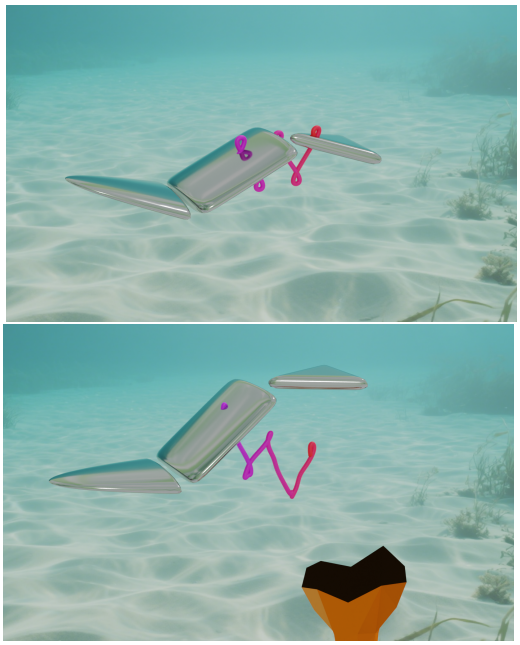


Figure 9: *The impact of a large oscillating fish tail on the robotic fish's movement, showing significant lateral drift caused by fluid interaction. The original and disturbed trajectories are shown in red for comparison.*

operators. Specifically, we compare three methods for solving Eq. 20, pseudo-inverse operator `torch.linalg.pinv`, least-squares solver `torch.linalg.lstsq`, and our custom operator `MinvF`. The evaluation is performed on a single-DOF object using meshes with varying face counts. As shown in Fig. 10, although `lstsq` offers a fast forward pass, its backward computation is slow, resulting in an overall performance similar to `pinv`. In contrast, our `MinvF` operator achieves substantial speedups over the native PyTorch methods, particularly for high-resolution meshes. To further demonstrate the efficiency of our approach, Table 1 presents a detailed breakdown of performance across all examples, including memory consumption and per-frame timings for forward computation, backpropagation, and other calculations.

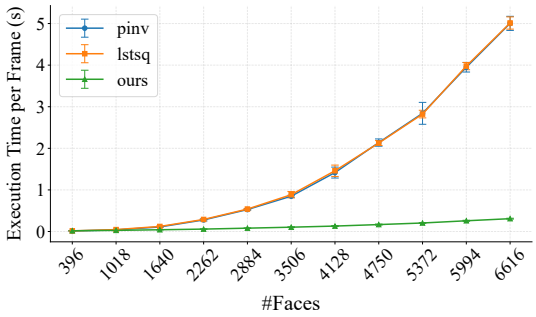


Figure 10: *Time consumption of different operators. All reported timings include complete forward and backward passes.*

Discussions While our assumption of potential flow limits the

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

Table 1: *Statistics. The column "Memory(MB)" records peak GPU memory consumption during runtime. The last four columns show the time consumption per frame for forward computation, backpropagation, other dynamics calculation parts, and the whole process, respectively.*

Case	#Rigid Bodies	DoF	#Faces	Memory(MB)	Forward (s)	Backward (s)	Others (ms)	Total Time(s)
Falling Ball	3	5	3582	2088	0.179	0.131	0.379	0.310
Grasping	4	6	3768	2475	0.187	0.225	0.546	0.413
Fish near Tail	4	6	5144	3821	0.382	0.436	0.587	0.819
Affected Clown Fish	4	8	4646	3877	0.395	0.544	0.700	0.940

range of phenomena our framework can capture, such as unsteady vortices or turbulence, it remains a good approximation in scenarios where turbulence plays only a minor role compared to inertial forces, such as slow or moderate motion of submerged bodies, like robotic grasping and object manipulation tasks, a regime commonly encountered in marine engineering [JLH*21, LYHJ22].

5. Conclusion

In conclusion, we have presented a framework that bridges the gap between efficiency and physical realism in underwater multi-rigid body simulations by extending the classical added mass paradigm through a fast BIE solver and variational dynamics principle. By capturing global fluid-mediated interactions without volumetric meshing and leveraging advanced preconditioning techniques, our method enables accurate and scalable simulation of complex multi-body dynamics in incompressible, irrotational fluids. Our approach not only advances the fidelity of underwater simulation, but also opens the door to practical deployment in underwater robotic applications.

Limitations. While our framework provides an efficient and physically grounded approach for simulating fluid-mediated interactions among multiple rigid bodies, it inherits limitations from the potential flow assumption. In particular, the irrotational model cannot capture vorticity, making it unsuitable for vortex-dominated phenomena such as wake interactions or turbulent flows. It would be interesting to incorporate approximate vorticity effects or hybrid schemes that blend potential flow with vortex particle methods. Such extensions could enable a broader range of flow conditions while maintaining advantages in speed and fidelity.

Acknowledgment

We thank Jiong Chen for his insightful comments in the early stages of this work.

References

[ACAT13] AKINCI N., CORNELIS J., AKINCI G., TESCHNER M.: Coupling elastic solids with smoothed particle hydrodynamics fluids. *Computer Animation and Virtual Worlds 24*, 3-4 (2013), 195–203. 2

[AIA*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile rigid-fluid coupling for incompressible sph. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 1–8. 2

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), 100–es. 1, 2, 7

[BGH03] BÖRM S., GRASEDYCK L., HACKBUSCH W.: Introduction to hierarchical matrices with applications. *Engineering analysis with boundary elements 27*, 5 (2003), 405–422. 2

[BSC14] BORG M., SHIRES A., COLLU M.: Offshore floating vertical axis wind turbines, dynamics modelling state of the art. part i: Aerodynamics. *Renewable and Sustainable Energy Reviews 39* (2014), 1214–1225. 3

[Car18] CARLTON J.: *Marine propellers and propulsion*. Butterworth-Heinemann, 2018. 3

[CB21] COUMANS E., BAI Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021. 6

[CM10] CHENTANEZ N., MÜLLER M.: Real-time simulation of large bodies of water with small scale details. In *Symposium on Computer Animation* (2010), pp. 197–206. 3

[CSB*19] CARPENTIER J., SAUREL G., BUONDONNO G., MIRABEL J., LAMIRAUX F., STASSE O., MANSARD N.: The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)* (2019). 5

[CSD24] CHEN J., SCHÄFER F. T., DESBRUN M.: Lightning-fast method of fundamental solutions. *ACM Transactions on Graphics 43*, 4 (2024), 77. 2, 3, 5, 7

[DHB*16] DA F., HAHN D., BATTY C., WOJTAN C., GRINSPUN E.: Surface-only liquids. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–12. 2

[Eri90] ERICKSON L. L.: *Panel methods: An introduction*. Tech. rep., 1990. 3

[GNS*12] GOLAS A., NARAIN R., SEWALL J., KRAJCEVSKI P., DUBEY P., LIN M.: Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 1–9. 2

[GSP*23] GROSS O., SOLIMAN Y., PADILLA M., KNÖPPEL F., PINKALL U., SCHRÖDER P.: Motion from shape change. *ACM Trans. Graph. 42*, 4 (2023), 107–1. 2, 3, 6

[GY15] GALLEGO G., YEZZI A.: A compact formula for the derivative of a 3-d rotation in exponential coordinates. *Journal of Mathematical Imaging and Vision 51* (2015), 378–384. 3

[HFG*18] HU Y., FANG Y., GE Z., QU Z., ZHU Y., PRADHANA A., JIANG C.: A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph. 37*, 4 (July 2018). 1, 2

[JLH*21] JIAO Y., LING F., HEYDARI S., HEESS N., MEREL J., KANSO E.: Learning to swim in potential flow. *Physical Review Fluids 6*, 5 (2021), 050505. 3, 8, 9

[JSMF*18] JESCHKE S., SKŘIVAN T., MÜLLER-FISCHER M., CHENTANEZ N., MACKLIN M., WOJTAN C.: Water surface wavelets. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–13. 3

[KMKM21] KETKAR N., MOOLAYIL J., KETKAR N., MOOLAYIL J.: Introduction to pytorch. *Deep learning with python: learn best practices of deep learning models with PyTorch* (2021), 27–91. 5

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

[KMRMH05] KANSO E., MARSDEN J. E., ROWLEY C. W., MELLI-HUBER J. B.: Locomotion of articulated bodies in a perfect fluid. *Journal of Nonlinear Science 15* (2005), 255–289. 2

[Lam24] LAMB H.: *Hydrodynamics*. University Press, 1924. 4

[LBH*22] LIU W., BAI K., HE X., SONG S., ZHENG C., LIU X.: Fishgym: A high-performance physics-based simulation framework for underwater robot learning. In *2022 International Conference on Robotics and Automation (ICRA)* (2022), IEEE Press, p. 6268–6275. 1

[LLDL21] LYU C., LI W., DESBRUN M., LIU X.: Fast and versatile fluid-solid coupling for turbulent flow simulation. *ACM Trans. Graph. 40*, 6 (Dec. 2021). 1, 2

[LSAM22] LEE H., SENGUPTA B., ARAGHIZADEH M., MYONG R.: Review of vortex methods for rotor aerodynamics and wake dynamics. *Advances in Aerodynamics 4*, 1 (2022), 20. 3

[LWP*23] LI W., WANG T., PAN Z., GAO X., WU K., DESBRUN M.: High-order moment-encoded kinetic simulation of turbulent flows. *ACM Transactions on Graphics (TOG) 42*, 6 (2023), 1–13. 6, 7

[LXY*23] LI Z., XU Q., YE X., REN B., LIU L.: Difffr: Differentiable sph-based fluid-rigid coupling for rigid body control. *ACM Trans. Graph. 42*, 6 (Dec. 2023). 1

[LYHJ22] LIN G., YANG Y., HE Z., JIAO P.: Hydrodynamic optimization in high-acceleration underwater motions using added-mass coefficient. *Ocean Engineering 263* (2022), 112274. 9

[MDZ*21] MA P., DU T., ZHANG J. Z., WU K., SPIELBERG A., KATZSCHMANN R. K., MATUSIK W.: Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–14. 1

[MWL*19] MIN S., WON J., LEE S., PARK J., LEE J.: Softcon: Simulation and control of soft-bodied animals with biomimetic actuators. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–12. 1, 3

[Not00] NOTAY Y.: Flexible conjugate gradients. *SIAM Journal on Scientific Computing 22*, 4 (2000), 1444–1460. 2

[OKRC10] OZGEN O., KALLMANN M., RAMIREZ L. E., COIMBRA C. F.: Underwater cloth simulation with fractional derivatives. *ACM Transactions on Graphics (TOG) 29*, 3 (2010), 1–9. 3

[PM18] PAN Z., MANOCHA D.: Active animations of reduced deformable models with environment interactions. *ACM Transactions on Graphics (TOG) 37*, 3 (2018), 1–17. 3

[RSCO25] ROMANYÀ-SERRASOLSAS M., CASAFRANCA J. J., OTADUY M. A.: Painless differentiable rotation dynamics. *Volume 44, Issue No. 4 of ACM Transactions on Graphics* (2025). 3

[SBH22] SUGIMOTO R., BATTY C., HACHISUKA T.: Surface-only dynamic deformables using a boundary element method. *Computer Graphics Forum 41*, 8 (2022), 75–86. 2

[SD06] STERN A., DESBRUN M.: Discrete geometric mechanics for variational time integrators. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), SIGGRAPH '06, Association for Computing Machinery, p. 75–80. 3

[SHW19] SCHRECK C., HAFNER C., WOJTAN C.: Fundamental solutions for water wave animation. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–14. 2

[SPG*24] SOLIMAN Y., PADILLA M., GROSS O., KNÖPPEL F., PINKALL U., SCHRÖDER P.: Going with the flow. *ACM Transactions on Graphics (TOG) 43*, 4 (2024), 1–12. 3, 6, 7

[SS86] SAAD Y., SCHULTZ M. H.: Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing 7*, 3 (1986), 856–869. 2, 5

[SSA*20] SHAHID T., SAJJAD F., AHSAN M., FARHAN S., SALAMAT S., ABBAS M.: Comparison of flow-solvers: Linear vortex lattice method and higher-order panel method with analytical and wind tunnel data. In *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)* (2020), IEEE, pp. 1–7. 3

[STW24] SU Z., TONG Y., WEI G.: Hodge decomposition of vector fields in cartesian grids. In *SIGGRAPH Asia 2024 Conference Papers* (2024), pp. 1–10. 4

[TGTL11] TAN J., GU Y., TURK G., LIU C. K.: Articulated swimming creatures. *ACM Trans. Graph. 30*, 4 (July 2011). 1

[WMS*23] WANG T.-H., MA P., SPIELBERG A. E., XIAN Z., ZHANG H., TENENBAUM J. B., RUS D., GAN C.: Softzoo: A soft robot co-design benchmark for locomotion in diverse environments. In *The Eleventh International Conference on Learning Representations* (2023). 1

[WP12] WEISSMANN S., PINKALL U.: Underwater rigid body dynamics. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 1–7. 2, 3, 4, 5

[XZX*23] XIAN Z., ZHU B., XU Z., TUNG H.-Y., TORRALBA A., FRAGKIADAKI K., GAN C.: Fluidlab: A differentiable environment for benchmarking complex fluid manipulation. *arXiv preprint arXiv:2303.02346* (2023). 1

[YBZ04] YING L., BIROS G., ZORIN D.: A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics 196*, 2 (2004), 591–626. 2

[YHK07] YUKSEL C., HOUSE D. H., KEYSER J.: Wave particles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007). 3

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics (TOG) 24*, 3 (2005), 965–972. 2

Appendix A: Notation

Table 2 summarizes symbols and notations used in this paper.

Table 2: *Notation table.*

Notation	Description
$\mathbf{T}_i \in \mathbb{R}^3$	Translation of body i
$\mathbf{R}_i \in \mathrm{SO}(3)$	Rotation of body i
$\theta_i \in \mathbb{R}^3$	Minimal 3D rotation vector for body i
$\mathbf{q}_i = (\mathbf{T}_i, \theta_i)$	Configuration of body i
$\mathbf{q}$	Configuration of the system
$\mathbf{M}(\mathbf{q})$	Generalized mass matrix
$V(\mathbf{q})$	User-defined potential energy
$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$	Lagrangian function
ω_i	Angular velocity of body i in world frame
$\mathbf{M}_{r,i}$	Generalized mass matrix of body i
$\mathbf{M}_{f,i}$	Decoupled fluid-added mass matrix of body i
m_i	Mass of body i
$\mathbf{I}_i \in \mathbb{R}^{3\times3}$	Inertia tensor of body i in body frame
ϕ	Potential field of the velocity field
$\Omega_i \subset \mathbb{R}^3$	Space taken by the ith rigid body in world frame
$\mathbf{n}_i(\mathbf{x})$	Outward normal of body i at $\mathbf{x}$ in world frame
$\mathcal{K}_r$	Kinetic energy of solid
$\mathcal{K}_f$	Kinetic energy of fluid
$\partial\Omega_i$	Boundary of body i
$\mathbf{x}_{ij}$	The world-space center of the jth face of the ith body
$\Phi(\mathbf{x}, \mathbf{y})$	3D fundamental solution of the Laplace equation
ρ	Fluid density
ϵ	Offset distance for source points
$\mathbf{K}_f$	Fundamental solution matrix
$\mathbf{K}_f^{\nabla}$	normal gradient matrix
$\mathbf{D}$	Diagonal matrix with face areas

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.