ICCFD13
MILAN ITALY 2026

Multi-Grid Accelerated Fluid-Particle-Interaction Simulations using the Immersed Boundary Method on GPU

J. Y. Ji* and C. A. Lin*
Corresponding author: calin@pme.nthu.edu.tw
* Department of Power Mechanical Engineering,
National Tsing Hua University, Hsinchu 30013, Taiwan

Abstract

A scalable numerical framework for incompressible fluid–structure interaction involving rigid bodies has been developed by coupling a sharp-interface immersed boundary method with a multigrid-accelerated fractional-step projection solver on distributed GPU platforms. The framework accurately captures complex FSI phenomena across a range of benchmark problems, including three-dimensional lid-driven cavity flow with an embedded rigid sphere and gravitational settling of up to 20,000 fully resolved particles. The results demonstrate the accuracy, robustness, and scalability of the proposed approach for large-scale simulations of particulate flows.

Keywords: particle-resolved, immersed boundary method, fluid-particle interaction, GPU.

1. Introduction

The immersed boundary method (IBM) was introduced, enabling fluid–solid coupling on fixed Cartesian grids and thereby eliminating the need for dynamic re-meshing. However, moving-boundary problems can cause spurious force oscillations (SFOs). To retain computational simplicity while suppressing SFOs, Liao et al. [1] introduced a velocity-interpolation-based forcing scheme that preserves rigid-body motion, achieves second-order spatial accuracy, and exhibits excellent numerical stability.

In incompressible flow simulations, the numerical treatment of pressure–velocity coupling plays a central role in determining both accuracy and computational efficiency. The most widely adopted approach is the projection method, in which a pressure Poisson equation is derived from the Navier–Stokes equations to enforce the divergence-free constraint [2, 3].

The elliptic pressure Poisson equation constitutes the principal computational bottleneck of the projection method. To accelerate its solution, multigrid algorithms (MG) [4, 5] are commonly employed, as they efficiently damp low-frequency errors through systematic transfer between coarse and fine grids. Chiu and Lin [6] demonstrated that increasing the number of grid levels significantly improves convergence efficiency. However, in distributed-memory parallel environments, domain decomposition restricts the achievable coarsening depth, thereby degrading multigrid performance. To overcome this limitation, coarse grid aggregation (CGA) [7] aggregates distributed subdomains into a unified coarse grid level, preserving multigrid efficiency for large-scale computations.

The resulting framework is demonstrated to be numerically robust and highly scalable through particle-laden FSI simulations involving up to 20,000 fully resolved three-dimensional particles, grid: $O(20^3)$ per particle.

2. Methodology

For incompressible viscous flows, the governing equations are given by the incompressible Navier–Stokes equations,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u}, \tag{1}$$

where $\boldsymbol{u}$ denotes the velocity vector, p is the pressure, ρ is the fluid density, and ν is the kinematic viscosity. The incompressibility constraint is enforced through the continuity equation,

$$\nabla \cdot \boldsymbol{u} \quad = \quad 0. \tag{2}$$

To compute the pressure, the classical projection method is adopted. A four-stage fractional-step scheme [8] is employed to advance the velocity field while maintaining numerical stability.

The first stage computes an intermediate velocity field using a total variation diminishing third-order Runge–Kutta (TVD-RK3) scheme [9]. The TVD-RK3 formulations are explicit and therefore well-suited for parallel implementation. However, the pressure Poisson equation is elliptic and cannot be solved explicitly. To efficiently resolve the pressure field within each physical time step, a semi-explicit dual-time stepping scheme is adopted [6],

$$\underbrace{\frac{1}{\beta}\frac{p^{k+1}-p^{k}}{\Delta\tau}+\frac{1}{\Delta t}\nabla\cdot\boldsymbol{u}^{**}-\frac{1}{\rho}\left(\nabla_{d}^{2}p^{k+1}+\nabla_{od}^{2}p^{k}\right)}_{L_{p}(p)}=0, \tag{3}$$

where k denotes the pseudo-time iteration index, $\Delta\tau$ is the pseudo-time step, and the pressure Laplacian is decomposed into diagonal and off-diagonal components evaluated at iterations $k+1$ and k, respectively. The artificial compressibility parameter β is set to 10^6 to accelerate convergence. Although the dual-time-stepping scheme enables robust enforcement of incompressibility, its convergence may be slow for large-scale problems. To improve efficiency, a multigrid acceleration technique is employed to solve the pressure Poisson equation[5, 6].

2.1 Multigrid acceleration

In iterative solvers for elliptic equations, high-frequency error components are typically damped rapidly, whereas low-frequency errors converge much more slowly and dominate the overall computational cost. The central idea of multigrid methods is to accelerate convergence by transferring low-frequency errors on fine grids to coarser grids, where they appear as higher-frequency components that can be eliminated more efficiently. In the present study, a V-cycle multigrid strategy is adopted, following the formulation of Shi et al. [5].

Restrictions and prolongations operations between adjacent grid levels are defined as

$$\tilde{p}^{h+1}=I_{h}^{h+1}p^{h},\qquad \hat{p}^{h}=I_{h+1}^{h}p^{h+1}, \tag{4}$$

where h denotes the multigrid level, with $h=0$ corresponding to the finest grid. The variable $\tilde{p}^{h+1}$ represents the restricted pressure field transferred from level h to level $h+1$, while $\hat{p}^{h}$ denotes the prolongated pressure field transferred from level $h+1$ back to level h.

The residual of the pressure Poisson equation at grid level h is defined as

$$R_{p}^{h}=\left(\frac{1}{\Delta t}\nabla\cdot\boldsymbol{u}^{**}-\frac{1}{\rho}\nabla^{2}p^{k+1}\right)^{h}. \tag{5}$$

On the coarse grid, the pressure Poisson equation is reformulated by incorporating the restricted approximate solution and residual as a source term, yielding

$$L_{p}^{h+1}\left(p^{h+1}\right)=\underbrace{L_{p_0}^{h+1}\left(\tilde{p}^{h+1}\right)-I_{h}^{h+1}\left(R_{p}^{h}-\overbrace{\mathrm{RHS}_{p}^{h}}^{\text{zero at } h=0}\right)}_{\mathrm{RHS}_{p}^{h+1}}, \tag{6}$$

where $L_{p}^{h+1}(\cdot)$ and $L_{p_0}^{h+1}(\cdot)$ are defined in equation (3). The right-hand side of equation (6) can be expanded as

$$\mathrm{RHS}_{p}^{h+1}=\left(\frac{1}{\Delta t}\nabla\cdot\boldsymbol{u}^{**}-\frac{1}{\rho}\nabla^{2}\tilde{p}^{k+1}\right)^{h+1}-I_{h}^{h+1}\left(R_{p}^{h}-\mathrm{RHS}_{p}^{h}\right). \tag{7}$$

When implementing multigrid methods on distributed GPU platforms, domain decomposition inherently limits the number of grid levels available on each processing unit, which can significantly

degrade multigrid efficiency. This limitation poses a major challenge for large-scale parallel simulations. To address this issue, a coarse-grid aggregation strategy (CGA) is employed [7]. The concept of CGA is illustrated in Fig. 1.

As an illustrative example, consider a discretized computational domain on a $32 \times 32 \times 32$ grid. On a single GPU, up to five multigrid levels can be employed. However, when the same domain is partitioned across a four-GPU cluster using domain decomposition in the z-direction, the maximum number of available grid levels is reduced to three, leading to degraded convergence performance. The CGA strategy mitigates this limitation by aggregating coarse-grid data from all GPUs onto a single processing unit at a prescribed grid level. This aggregation enables the continuation of the multigrid process with the same effective number of grid levels as in the single-GPU case, thereby preserving convergence efficiency.

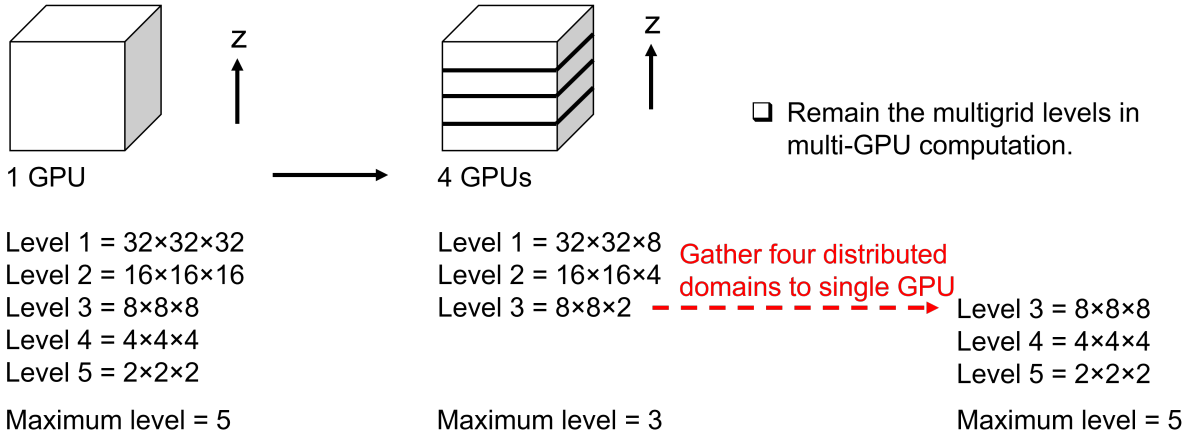


Figure 1: Schematic illustration of the coarse-grid aggregation (CGA) strategy.

The V-cycle multigrid procedure is adopted in the present study. The algorithm consists of a downward traversal through successively coarser grids via restriction, followed by an upward traversal through finer grids via prolongation. When the number of grid points falls below a prescribed threshold, the CGA technique is activated to maintain multigrid efficiency.

The complete multigrid algorithm follows a V-cycle structure and consists of three principal stages: pre-smoothing on the finest grid, recursive coarse-grid correction with restriction and prolongation operations, and post-smoothing to finalize the pressure solution. The detailed procedure is summarized as follows:

1. *Pre-smoothing:*
   At the finest grid level ($h = 0$), the pressure field p^h is obtained by solving
   $$L_p^h(p^h) = 0$$
   using α_1 smoothing iterations.
2. *Coarse-grid correction:*
   (a) Restrict the approximate pressure solution and residual from level h to $h + 1$:
   $$\tilde{p}^{h+1} = I_h^{h+1} p^h, \qquad \tilde{R}^{h+1} = I_h^{h+1} R^h.$$
   (b) Compute the coarse-grid right-hand side RHS_p^{h+1} using equation (6).
   (c) Solve the coarse-grid equation
   $$L_p^{h+1}(p^{h+1}) = \mathrm{RHS}_p^{h+1}$$
   using α_{rs} smoothing iterations.
   (d) Increment grid level: $h = h + 1$.
   (e) If $h = h_{\text{threshold}}$, perform CGA via MPI_Allgather so that all GPUs possess identical coarse-grid data, and return to step (a). If h corresponds to the coarsest grid level, proceed to step (f); otherwise, return to step (a).

(f) Decrement grid level: $h = h - 1$.

(g) Correct the pressure solution at level h:

$$p^h = p^h + I_{h+1}^h \left(p^{h+1} - \tilde{p}^{h+1} \right).$$

(h) Apply α_{pl} post-smoothing iterations:

$$L_p^h(p^h) = \mathrm{RHS}_p^h.$$

(i) If $h - 1$ is the finest level:

i. Set $h = h - 1$.

ii. Update the pressure field:

$$p^h = p^h + I_{h+1}^h \left(p^{h+1} - \tilde{p}^h \right).$$

iii. Proceed to step 3.

Otherwise, return to step (f).

3. *Post-smoothing:*
At the finest grid level ($h = 0$), the pressure field is finalized by solving

$$L_p^h(p^h) = 0$$

using α_1 smoothing iterations.

The parameters α_1, α_{rs}, and α_{pl} denote the number of iterations employed during pre-smoothing, coarse-grid smoothing, and post-smoothing, respectively. The choice of these parameters plays a critical role in determining the overall efficiency and convergence behavior of the multigrid solver. In the present study, the iteration settings follow the configuration proposed by Chiu and Lin [6].

This multigrid–CGA strategy enables efficient and scalable solution of the pressure Poisson equation in distributed-memory environments without degrading convergence at coarse levels.

2.2 Immersed boundary method

In the immersed boundary method, the momentum equations and the projection step are solved on the entire computational domain using a fixed Cartesian grid, without explicitly distinguishing between fluid and solid regions in the discretization. The effect of rigid boundaries is imposed through an additional body-force term that enforces the desired kinematic constraint. In the present study, the velocity-interpolation-based forcing method proposed by Liao et al. [1] is adopted to evaluate this forcing term.

Before computing the immersed boundary force, computational nodes are classified as fluid, forcing, or solid nodes according to the staggered-grid arrangement. Taking the u-velocity nodes as an example (Fig. 2), nodes located entirely inside the rigid body are classified as solid nodes (green triangles). A node is identified as a forcing node (red triangles) if it has at least one neighboring solid node in the x- or y-direction; the immersed boundary force is applied at these locations. All remaining nodes are classified as fluid nodes (blue triangles).

Once the node classification is completed, the body-force term is evaluated. In the velocity interpolation method [1], the forcing applied at each forcing node is computed as

$$\boldsymbol{f}^{n+1} = \frac{\boldsymbol{u}_F - \widehat{\boldsymbol{u}}}{\Delta t}. \tag{8}$$

In Eq. (8), $\widehat{\boldsymbol{u}}$ denotes the predicted velocity obtained by advancing the momentum equation without the body-force term,

$$\widehat{\boldsymbol{u}} = \boldsymbol{u}^n + \Delta t \left(-\nabla\cdot(\boldsymbol{u}^n\boldsymbol{u}^n) - \frac{1}{\rho}\nabla p^n + \nu\nabla^2\boldsymbol{u}^n \right). \tag{9}$$

The target velocity at the forcing node, $\boldsymbol{u}_F$, is obtained by interpolating between a neighboring fluid node and the rigid-body velocity at the immersed boundary. The procedure is illustrated in Fig. 2. As shown in Fig. 2(a), nodes A, F, and B lie on the same vertical line, where A denotes a fluid node, F is the forcing node, and B is the intersection point between the grid line and the immersed

boundary. As shown in Fig. 2(b), $\boldsymbol{u}_F$ is calculated by linear interpolation between the predicted velocity at node A, $\widehat{\boldsymbol{u}}_A$, and the rigid-body velocity at point B, denoted by $\boldsymbol{U}_{move}$:

$$\boldsymbol{u}_F = \widehat{\boldsymbol{u}}_A + \frac{\overline{FA}}{\overline{AB}}\boldsymbol{U}_{move} - \widehat{\boldsymbol{u}}_A \,. \tag{10}$$

The linear interpolation for $\boldsymbol{u}_F$ achieves second-order accuracy. The overall spatial accuracy of the sharp-interface immersed boundary formulation has been verified to be second-order in our previous work [1].

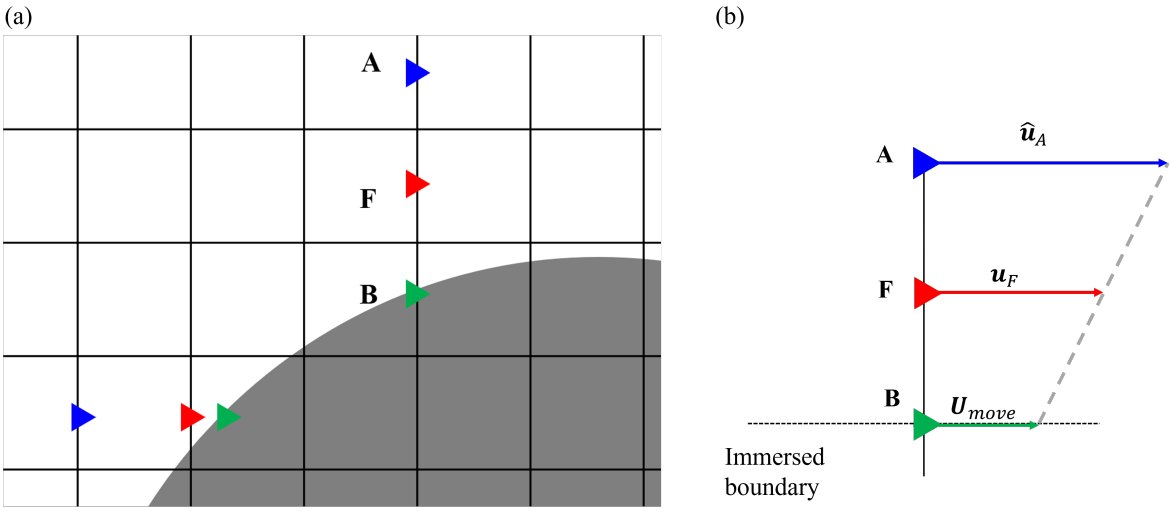


Figure 2: Schematic of the velocity-interpolation forcing method: (a) node locations; (b) velocity interpolation between the neighboring fluid node and the boundary point.

With this formulation, the immersed boundary force modifies the predicted velocity so that the updated velocity at the forcing node matches the interpolated target velocity in the subsequent time step. In addition to the velocity interpolation method, the solid-body forcing (SBF) approach [1] is also employed. Unlike the forcing-node treatment above, the SBF method applies forcing directly on solid nodes. The forcing term is defined as

$$\boldsymbol{f}^{n+1} = \frac{\boldsymbol{U}_{move} - \widehat{\boldsymbol{u}}}{\Delta t} \,. \tag{11}$$

This expression is analogous to the velocity interpolation method, except that the target velocity is directly prescribed as the rigid-body velocity.

2.3 Rigid-body motion

Rigid-body motion is advanced using a forward Euler scheme. The translational velocity and position of the particle center are updated as

$$\mathbf{v}_s^{n+1} = \mathbf{v}_s^n + \Delta t\, \mathbf{F}_{net}/m_s, \tag{12}$$

$$\mathbf{x}_s^{n+1} = \mathbf{x}_s^n + \mathbf{v}_s^{n+1}\Delta t. \tag{13}$$

The angular velocity is updated according to

$$\boldsymbol{\omega}_s^{n+1} = \boldsymbol{\omega}_s^n + \frac{\Delta t}{I_s}\mathbf{T}_{\text{net}}, \tag{14}$$

where m_s and I_s denote the particle mass and moment of inertia, respectively. The hydrodynamic torque acting on a particle is computed from the immersed boundary forcing as

$$\mathbf{T}_{\text{net}} = \mathbf{T}_{\text{fluid}} = -\rho_f \int_s \mathbf{r} \times \mathbf{f}\, dV, \tag{15}$$

where $\mathbf{r}$ is the position vector from the particle center to the Lagrangian forcing point, ρ_f is the fluid

density, and $\mathbf{f}$ is the immersed boundary forcing term.

The net force acting on the solid body, $\mathbf{F}_{net}$, is defined as

$$\mathbf{F}_{net} = \mathbf{F}_{fluid} + \mathbf{F}_G + \mathbf{F}_B + \mathbf{F}_{collision}, \tag{16}$$

where $\mathbf{F}_{fluid}$ denotes the total hydrodynamic force exerted by the surrounding fluid. It is obtained by integrating the immersed boundary force over the solid volume,

$$\mathbf{F}_{fluid} = -\rho_f \int_s \mathbf{f}\, dV.$$

Here, ρ_f is the fluid density and $\mathbf{f}$ is the immersed boundary forcing term.

The combined gravitational and buoyancy force can be expressed as

$$\mathbf{F}_G + \mathbf{F}_B = (\rho_s - \rho_f) g V, \tag{17}$$

where ρ_s denotes the density of the solid, g is the gravitational acceleration, and V is the volume of the rigid body.

The term $\mathbf{F}_{collision}$ accounts for inter-particle collisions. In this study, the collision model proposed by Glowinski et al. [10] is adopted. The collision force depends solely on the inter-particle gap distance and can be interpreted as a nonlinear spring model. The force is activated only when the gap falls below a prescribed threshold. During the approach phase, the gap decreases and the repulsive force increases rapidly, thereby preventing particle overlap. Conversely, during separation, the gap increases and the force decreases smoothly and continuously, vanishing as the gap exceeds the threshold. The particle–particle collision force is given by

$$\mathbf{F}_{collision} = \begin{cases} 0, & \left\|\mathbf{x}_i - \mathbf{x}_j\right\| > R_i + R_j + \xi, \\ \dfrac{c_{ij}}{\varepsilon}\left(\dfrac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\| - R_i - R_j - \xi}{\xi}\right)^2 \left(\dfrac{\mathbf{x}_i - \mathbf{x}_j}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|}\right), & \left\|\mathbf{x}_i - \mathbf{x}_j\right\| \le R_i + R_j + \xi, \end{cases} \tag{18}$$

where c_{ij} is the force scaling factor, defined as $m_s g$ (i.e., the gravitational force acting on a solid particle), and ε is set to 2×10^{-3}. This value follows the parameter choice reported in [11], where it was employed in simulations of two-particle settling and demonstrated good agreement with reference results. The same parameter value is adopted in the present study for simulations involving 20,000 particles, where it likewise produces satisfactory results. The vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ represent the particle-center positions of particles i and j, respectively, and R_i and R_j denote their radii. The parameter ξ defines the activation range of the collision model and is set equal to the grid spacing.

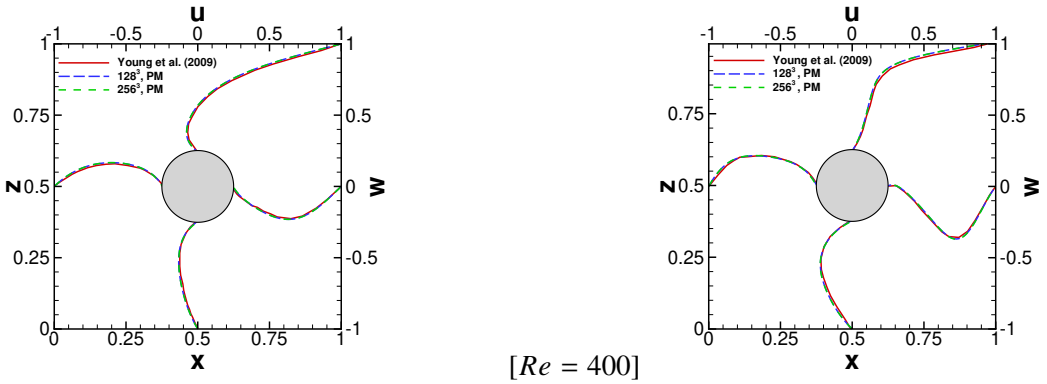
3. Results

Simulations were performed on distributed GPU platforms using 4–32 NVIDIA V100 GPUs. The validation of the proposed computational framework begins with a three-dimensional lid-driven cavity flow containing a centrally embedded sphere. The study then considers prescribed-motion benchmarks to further assess the robustness of the coupled projection and immersed boundary formulation. Finally, the solver is applied to particle sedimentation under gravity, progressing from single-particle settling to two-particle interactions and, ultimately, to large-scale simulations involving up to 20,000 particles.

The first benchmark considers a three-dimensional lid-driven cavity flow with an embedded sphere. The computed centerline u- and w-velocity profiles at $Y = 0.5$ agree well with the reference data reported by Young et al. [12]. Moreover, the two grid resolutions produce nearly indistinguishable profiles, indicating adequate mesh convergence for the present configuration.

The computational performance for different mesh resolutions is summarized in Fig. 4, including the results using the general pressure equation[13]. Overall, the projection method is substantially more efficient than the GPE approach for both configurations. For the standard cavity case, the projection method is approximately 1.6–9 times faster, and the speedup increases to roughly 3–16 when the embedded sphere is present. These results highlight the computational advantage of the multigrid-accelerated projection framework for incompressible flows involving complex immersed

geometries.



[Re = 100] [Re = 400]

Figure 3: Lid-driven cavity flow with an embedded sphere: centerline velocity profiles compared with Young et al. [12].

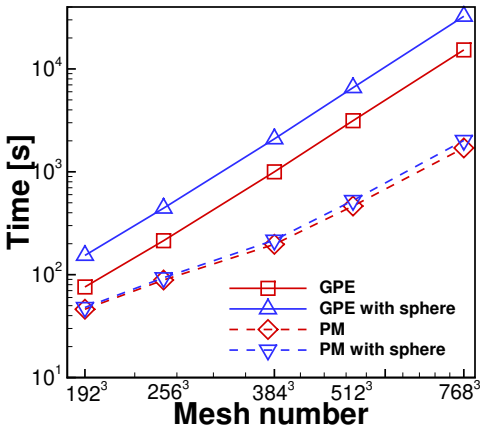


Figure 4: Computational time comparison for the lid-driven cavity benchmark.

To further evaluate scalability and large-scale applicability, a simulation of 20,000 particles settling under gravity is performed. The fluid and particle properties are identical to those used in the two-particle case. To reduce computational cost, the particle resolution is reduced to $D = 21.3\Delta x$. The computational domain spans $24D \times 24D \times 96D$ and is discretized on a $512 \times 512 \times 2048$ grid, resulting in 5.37×10^8 grid points. The particle volume fraction is 18.94%.

Figure 5 shows the temporal evolution of the settling process. Particles near the corners descend more rapidly, producing finger-like structures reminiscent of Rayleigh–Taylor instability. The associated vortical structures are visualized in Fig. 6 using the Q-criterion ($Q = 0.1$). A distinct upward jet-like plume develops at approximately $t \approx 6.25$ s, accompanied by coherent vortex structures. These results demonstrate that the proposed solver can efficiently resolve large-scale particle-laden flows and capture complex collective dynamics in dense suspensions.

4. Conclusions

A scalable numerical framework for the interaction of incompressible fluids and structures (FSI) involving rigid bodies has been developed by coupling a sharp-interface immersed boundary formulation with a multigrid-accelerated fractional-step projection method. The solver is implemented on distributed GPU platforms to enable large-scale simulations, while the underlying numerical formulation remains general and independent of hardware-specific assumptions.

The accuracy and robustness of the proposed framework were verified through a hierarchy of benchmark problems. In the three-dimensional lid-driven cavity flow with an embedded rigid sphere, the solver accurately reproduced the modification of the primary recirculation structure and its systematic evolution with increasing Reynolds number. The framework was also applied to large-

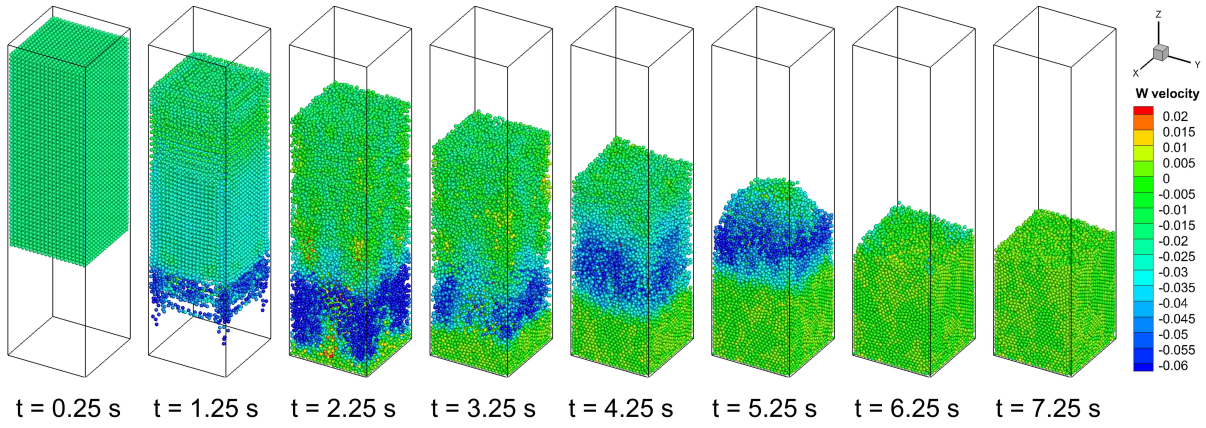


Figure 5: Time evolution of the settling process for 20,000 particles.

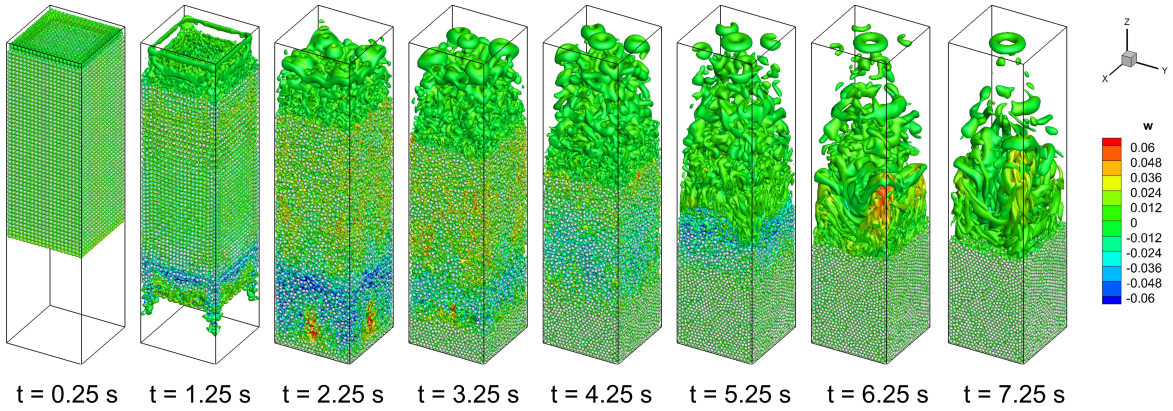


Figure 6: Vortical structures for the 20,000-particle settling case visualized using the Q-criterion ($Q = 0.1$).

scale gravitational settling simulations involving up to 20,000 fully resolved particles, revealing complex flow instabilities and coherent vortex structures in dense particulate systems.

References

[1] Chuan-Chieh Liao, Yu-Wei Chang, Chao-An Lin, and JM McDonough. Simulating flows with moving rigid boundary using immersed-boundary method. *Computers & Fluids*, 39(1):152–167, 2010.

[2] Alexandre Joel Chorin. Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762, 1968.

[3] John Kim and Parviz Moin. Application of a fractional-step method to incompressible navier-stokes equations. *Journal of computational physics*, 59(2):308–323, 1985.

[4] Achi Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977.

[5] Xiaolei Shi, Tanmay Agrawal, Chao-An Lin, Feng-Nan Hwang, and Tzu-Hsuan Chiu. A parallel nonlinear multigrid solver for unsteady incompressible flow simulation on multi-gpu cluster. *Journal of Computational Physics*, 414:109447, 2020.

[6] Tzu-Hsuan Chiu and Chao-An Lin. Multigrid accelerated projection method on gpu cluster for the simulation of turbulent flows. *Journal of Mechanics*, 39:199–212, 2023.

[7] Kengo Nakajima. Openmp/mpi hybrid parallel multigrid method on fujitsu fx10 supercomputer system. In *2012 IEEE International Conference on Cluster Computing Workshops*, pages 199–206. IEEE, 2012.

[8] Haecheon Choi and Parviz Moin. Effects of the computational time step on numerical solutions of turbulent flow. *Journal of Computational Physics*, 113(1):1–4, 1994.

[9] John H Williamson. Low-storage runge-kutta schemes. *Journal of computational physics*, 35(1):48–56, 1980.

[10] Roland Glowinski, T-W Pan, Todd I Hesla, and Daniel D Joseph. A distributed lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25(5):755–794, 1999.

[11] Chuan-Chieh Liao, Wen-Wei Hsiao, Tingyu Lin, and Chao-An Lin. Simulations of two sedimenting-interacting spheres with different sizes and initial configurations using immersed boundary method. *Computational Mechanics*, 55:1191–1200, 2015.

[12] DL Young, YC Lin, CM Fan, and CL Chiu. The method of fundamental solutions for solving incompressible navier–stokes problems. *Engineering analysis with boundary elements*, 33(8-9):1031–1044, 2009.

[13] Xiaolei Shi and Chao-An Lin. Simulations of wall bounded turbulent flows using general pressure equation. *Flow, Turbulence and Combustion*, 105(1):67–82, 2020.