White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Leeds.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/246353/

Version: Accepted Version

Article:
Chen, M., Wang, Y. and Jimack, P.K. (2026) A Parallel One-Field Fictitious-Domain Finite Element Method for Large-Scale Fluid–Structure Interaction. International Journal for Numerical Methods in Fluids. ISSN: 0271-2091

https://doi.org/10.1002/fld.70107

This is an author produced version of an article published in International Journal for Numerical Methods in Fluids, made available via the University of Leeds Research Outputs Policy under the terms of the Creative Commons Attribution License (CC-BY), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

A Parallel One-Field Fictitious-Domain Finite Element Method for Large-Scale Fluid–Structure Interaction

Meng-Huo Chen[1,*], Yongxing Wang[2], Peter K. Jimack[2]

[1]Department of Mathematics, National Chung Cheng University, Chiayi 621301, Taiwan
[2]School of Computing, University of Leeds, Woodhouse, Leeds LS2 9JT, United Kingdom

[*]Corresponding author: menghuo@ccu.edu.tw

Author Accepted Manuscript

Abstract

We present a distributed-memory finite-element framework for large-scale incompressible fluid–structure interaction (FSI) based on a monolithic one-field fictitious-domain formulation. The velocity and pressure are defined on a fixed Eulerian background mesh and discretized using Taylor–Hood finite elements with quadratic velocity and linear pressure approximation, while the immersed solid is represented by an updated Lagrangian mesh. The resulting saddle-point systems are solved by preconditioned MINRES with a symmetric split preconditioner, which takes a block-Jacobi form in parallel. The method is assessed using two- and three-dimensional lid-driven-cavity benchmarks involving deformable discs, slabs, spheres, and cylinders. Serial–parallel comparisons, temporal and background-mesh refinement studies, and quantitative solid-motion measures support the numerical consistency of the implementation. Strong-scaling results are reported up to 256 MPI ranks in two dimensions and 512 MPI ranks in three dimensions. Matrix assembly scales well, whereas degradation of the block-Jacobi preconditioner is the main factor limiting linear-solver efficiency. Nevertheless, the parallel implementation substantially reduces wall-clock time and enables large three-dimensional monolithic FSI simulations that would be prohibitively expensive in serial.

Keywords: Fluid–structure interaction; finite element method; parallel computing; fictitious-domain method; monolithic formulation

1 Introduction

Fluid–structure interaction (FSI) arises in many problems in fluid mechanics, biomechanics, and engineering, where the motion of a fluid and the deformation of a solid mutually influence one another. Representative examples include blood flow in compliant arteries and heart valves, airflow–tissue interaction in the respiratory system, and cerebrospinal fluid dynamics in deformable brain compartments [1]; biological locomotion such as swimming and flying organisms [2]; vortex-induced vibration and flutter of slender structures and cables in wind and ocean engineering [3, 4]; and aeroelastic coupling of wings, turbine blades, and control surfaces in aerospace applications [3]. FSI also appears in fluid-driven actuators, soft robotic components, membrane filtration, and microfluidic devices in which flexible boundaries can strongly affect the surrounding flow [5]. These applications often involve moving interfaces, large deformation, incompressibility constraints, and wide ranges of material parameters, making robust and efficient numerical simulation challenging, particularly in three dimensions.

Numerical FSI methods are commonly classified into *partitioned* and *monolithic* approaches. In partitioned methods [6, 7], the fluid and structural subproblems are advanced separately

and coupled through interface conditions. This strategy is modular and allows the reuse of existing solvers, but it may suffer from added-mass instabilities or slow convergence in strongly coupled regimes unless suitable stabilization, relaxation, or subiteration techniques are used. In monolithic methods [2, 8, 9], the coupled fluid and solid unknowns are solved simultaneously. Such methods are generally more robust for strongly coupled FSI problems, but they lead to larger and more difficult algebraic systems, often with indefinite saddle-point structure. These trade-offs have been discussed in several reviews and comparative studies; see, for example, Hou et al. [10] and Michler et al. [11].

Another important issue is the treatment of the moving fluid–solid interface. Body-fitted moving-mesh methods can represent the interface accurately, but mesh motion or remeshing may become expensive or fragile when the deformation is large. Fictitious-domain and immersed methods avoid this difficulty by embedding the solid in a fixed background fluid mesh and representing the solid motion on a separate Lagrangian mesh. Representative approaches include the immersed boundary method [12] and the fictitious-domain mortar formulation of Baaijens [13], in which the fluid–solid kinematic coupling is imposed using Lagrange multipliers. Such approaches are attractive for large-deformation FSI because they avoid repeated remeshing of the fluid domain while retaining fixed Eulerian data structures for the fluid computation.

The present work considers a *one-field monolithic fictitious-domain* formulation for incompressible FSI. In this formulation, the fluid and solid equations are written over a common computational domain, with a single global velocity and pressure field. The immersed solid is represented by an updated Lagrangian mesh, and its inertial and elastic contributions enter the weak form through correction terms over the current solid region. The coupling between the background mesh and the solid mesh is realized by interpolation between the two finite-element representations. This construction enforces the fluid–solid coupling without explicit interface reconstruction and is compatible with standard finite-element discretizations for incompressible flow. A representative one-field monolithic fictitious-domain formulation for general FSI was developed by Wang, Jimack, and Walkley [14].

Although this formulation is attractive for moving-interface problems, its large-scale application depends strongly on the efficiency of the resulting linear algebra. After temporal discretization and Taylor–Hood finite-element approximation, the diffusion–pressure stage leads to coupled saddle-point systems of Navier–Stokes type, augmented by solid inertial and elastic contributions. In three dimensions, these systems rapidly become too large for serial computation. Moreover, monolithic FSI systems are often more ill-conditioned than the corresponding single-physics fluid systems, and their efficient solution requires effective preconditioning. Block preconditioners, Schur-complement approximations, and physics-based splitting strategies have therefore played an important role in the development of scalable monolithic FSI solvers [9, 15].

The objective of this paper is to develop and assess a distributed-memory parallel implementation of the one-field monolithic fictitious-domain method for large-scale FSI simulation. The background fluid domain is discretized by Taylor–Hood elements, with quadratic velocity and linear pressure approximation, while the immersed solid is represented by a Lagrangian finite-element mesh. The solid-to-fluid coupling is assembled through an interpolation operator between the two meshes, leading at each time step to a monolithic saddle-point system for the global velocity and pressure unknowns. The resulting systems are solved by preconditioned MINRES with symmetric split preconditioning. In the parallel implementation, the preconditioner is applied in a block–Jacobi manner over the subdomains. This provides a simple and practical baseline preconditioner for distributed-memory computation, but it also weakens the preconditioner by neglecting off-process couplings.

The method is tested on two- and three-dimensional lid-driven-cavity FSI benchmarks involving deformable discs, slabs, spheres, and cylinders. We report strong-scaling results up to 256 MPI ranks in two dimensions and 512 MPI ranks in three dimensions. In addition to total wall-clock time, we separate the costs of matrix assembly and linear-system solution and

examine the dependence of PMINRES iteration counts on the domain decomposition. The results show that the proposed implementation substantially reduces the computational cost of monolithic fictitious-domain FSI simulation and makes three-dimensional computations feasible at resolutions that would be prohibitively expensive in serial. They also show that the finite-element assembly scales well, whereas the block–Jacobi approximation of the preconditioner is the main factor limiting linear-solver scalability. Thus, the contribution of this work is a distributed-memory realization of a monolithic one-field fictitious-domain FSI method together with a quantitative assessment of the solver bottlenecks that must be addressed for more scalable large-scale FSI simulations.

2 Governing Equations

We consider a fluid–structure interaction (FSI) problem in a bounded spatial domain $\Omega \subset \mathbb{R}^d$ ($d = 2$ or 3) over the time interval $(0, T]$, as illustrated in Fig. 1. At each time $t \in (0, T]$, the domain is partitioned into a fluid region Ω_t^f and a solid region Ω_t^s such that

$$\overline{\Omega} = \overline{\Omega_t^f} \cup \overline{\Omega_t^s}, \qquad \Omega_t^f \cap \Omega_t^s = \emptyset,$$

with moving interface

$$\Gamma_t = \partial\Omega_t^f \cap \partial\Omega_t^s.$$

Let $\boldsymbol{u}^f(\boldsymbol{x}, t)$ and $p^f(\boldsymbol{x}, t)$ denote the fluid velocity and pressure in Ω_t^f, and let $\boldsymbol{u}^s(\boldsymbol{x}, t)$ and $p^s(\boldsymbol{x}, t)$ denote the solid velocity and Lagrange multiplier (pressure) in Ω_t^s. The fluid density and dynamic viscosity are denoted by ρ_f and μ_f, respectively, while ρ_s and μ_s denote the solid density and shear modulus. The outward unit normals on $\partial\Omega_t^f$ and $\partial\Omega_t^s$ are denoted by $\boldsymbol{n}^f$ and $\boldsymbol{n}^s$, respectively. Throughout, indices $i, j, k \in \{1, \ldots, d\}$ indicate spatial components, and the repeated-index summation convention is employed.

Fluid equations. In the fluid region Ω_t^f, the incompressible Navier–Stokes equations in Eulerian form are

$$\frac{\partial u_j^f}{\partial x_j} = 0, \tag{1}$$

$$\rho_f \frac{Du_i^f}{Dt} - \frac{\partial \sigma_{ij}^f}{\partial x_j} = \rho_f g_i, \tag{2}$$

$$\sigma_{ij}^f = \mu_f \left(\frac{\partial u_i^f}{\partial x_j} + \frac{\partial u_j^f}{\partial x_i} \right) - p^f \delta_{ij} = \tau_{ij}^f - p^f \delta_{ij}, \tag{3}$$

where $\boldsymbol{g}$ is the body force per unit mass, δ_{ij} is the Kronecker delta, and τ_{ij}^f denotes the deviatoric fluid stress.

Solid equations. In the solid region Ω_t^s, we adopt an incompressible neo-Hookean model written in the current configuration:

$$\frac{\partial u_j^s}{\partial x_j} = 0, \tag{4}$$

$$\rho_s \frac{Du_i^s}{Dt} - \frac{\partial \sigma_{ij}^s}{\partial x_j} = \rho_s g_i, \tag{5}$$

$$\sigma_{ij}^s = \mu_s \left(F_{ik} F_{jk} - \delta_{ij} \right) - p^s \delta_{ij} = \tau_{ij}^s - p^s \delta_{ij}, \tag{6}$$

where $\mathbf{F} = [F_{ij}]$ is the deformation gradient with

$$F_{ij} = \frac{\partial x_i}{\partial X_j},$$

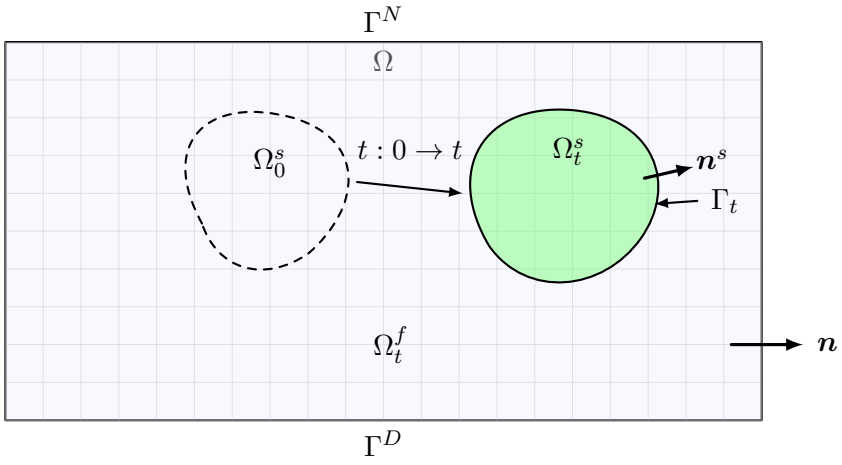


Figure 1: Schematic of a fluid–structure interaction configuration. The fixed computational domain Ω is decomposed into the fluid region Ω_t^f and the time-dependent solid region Ω_t^s. The dashed curve indicates the reference solid configuration Ω_0^s, while Γ_t denotes the moving fluid–solid interface. The lower and upper outer boundaries illustrate Dirichlet and Neumann boundary portions, respectively.

$\boldsymbol{X}$ and $\boldsymbol{x}$ denote the reference and current coordinates, respectively, and $\tau_{ij}^s = \mu_s(F_{ik}F_{jk} - \delta_{ij})$ is the deviatoric Cauchy stress of the solid.

The incompressibility constraints (1) and (4) are enforced through the pressures p^f and p^s in the fluid and solid subdomains, respectively.

Interface conditions. Across the moving interface Γ_t, the kinematic and dynamic coupling conditions are

$$u_i^f = u_i^s \qquad \text{on } \Gamma_t, \tag{7}$$

$$\sigma_{ij}^f n_j^s = \sigma_{ij}^s n_j^s \qquad \text{on } \Gamma_t. \tag{8}$$

The first condition enforces continuity of velocity, while the second enforces continuity of traction.

Boundary conditions. On the exterior boundary $\partial\Omega$, we assume a partition

$$\partial\Omega = \Gamma^D \cup \Gamma^N, \qquad \Gamma^D \cap \Gamma^N = \emptyset.$$

The boundary conditions for the unified velocity and stress on the exterior boundary are taken as

$$u_i^f = \bar{u}_i \qquad \text{on } \Gamma^D, \tag{9}$$

$$\sigma_{ij}^f n_j = h_i \qquad \text{on } \Gamma^N, \tag{10}$$

where $\bar{\boldsymbol{u}}$ is the prescribed boundary velocity and $\boldsymbol{h}$ is the prescribed traction. For the configuration shown in Fig. 1, the lower boundary is an example of Γ^D and the upper boundary is an example of Γ^N.

3 Discretization of the governing equations

We next describe the temporal and spatial discretization of the coupled FSI problem. Starting from the weak statements associated with (1), (2), (4), and (5), we introduce the L^2 inner product

$$(u, w)_\Omega := \int_\Omega uw \, d\Omega,$$

and define the unified velocity and pressure fields by

$$u_i = \begin{cases} u_i^f, & \text{in } \Omega_t^f, \\ u_i^s, & \text{in } \Omega_t^s, \end{cases} \qquad p = \begin{cases} p^f, & \text{in } \Omega_t^f, \\ p^s, & \text{in } \Omega_t^s. \end{cases} \tag{11}$$

Using the constitutive relations (3) and (6), together with the traction boundary condition (10), integration by parts yields a one-field variational formulation on the fixed background domain Ω, supplemented by correction terms over the current solid subdomain Ω_t^s; see also [14]. More precisely, for test functions $v_i \in H_0^1(\Omega)$ and $q \in L_0^2(\Omega)$, we seek $(u_i, p) \in H^1(\Omega) \times L_0^2(\Omega)$ such that

$$\begin{aligned} &\rho_f \left(\frac{Du_i}{Dt}, v_i \right)_\Omega + \left(\tau_{ij}^f, \frac{\partial v_i}{\partial x_j} \right)_\Omega - \left(p, \frac{\partial v_j}{\partial x_j} \right)_\Omega - \left(\frac{\partial u_j}{\partial x_j}, q \right)_\Omega \\ &\quad + (\rho_s - \rho_f) \left(\frac{Du_i}{Dt}, v_i \right)_{\Omega_t^s} + \left(\tau_{ij}^s - \tau_{ij}^f, \frac{\partial v_i}{\partial x_j} \right)_{\Omega_t^s} \\ &= (\bar{h}_i, v_i)_{\Gamma^N} + \rho_f (g_i, v_i)_\Omega + (\rho_s - \rho_f)(g_i, v_i)_{\Omega_t^s}. \end{aligned} \tag{12}$$

A key feature of (12) is that the background-fluid and solid contributions are evaluated using different mesh descriptions. The background domain Ω is represented by a fixed Eulerian mesh, whereas the immersed solid is described by an updated Lagrangian mesh. Accordingly, the material derivative is expressed as

$$\frac{Du_i}{Dt} = \frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \qquad \text{in } \Omega, \tag{13}$$

whereas, following the motion of the Lagrangian solid,

$$\frac{Du_i^s}{Dt} = \frac{\partial u_i^s}{\partial t} \qquad \text{in } \Omega_t^s. \tag{14}$$

Using a first-order time discretization, (12) is written at time level t^{n+1} as

$$\begin{aligned} &\rho_f \left(\frac{u_i - u_i^n}{\Delta t} + u_j \frac{\partial u_i}{\partial x_j}, v_i \right)_\Omega + \left(\tau_{ij}^f, \frac{\partial v_i}{\partial x_j} \right)_\Omega - \left(p, \frac{\partial v_j}{\partial x_j} \right)_\Omega - \left(\frac{\partial u_j}{\partial x_j}, q \right)_\Omega \\ &\quad + (\rho_s - \rho_f) \left(\frac{u_i - u_i^n}{\Delta t}, v_i \right)_{\Omega_{n+1}^s} + \left(\tau_{ij}^s - \tau_{ij}^f, \frac{\partial v_i}{\partial x_j} \right)_{\Omega_{n+1}^s} \\ &= (\bar{h}_i, v_i)_{\Gamma^N} + \rho_f (g_i, v_i)_\Omega + (\rho_s - \rho_f)(g_i, v_i)_{\Omega_{n+1}^s}, \end{aligned} \tag{15}$$

where Ω_{n+1}^s denotes the current solid domain at the new time level.

To advance the solution in time, we follow the operator-splitting approach commonly used for incompressible flow problems [16, 17]. Equation (15) is decomposed into a convection stage and a diffusion–pressure stage:

1. **Convection step**

$$\rho_f \left(\frac{u_i^\star - u_i^n}{\Delta t} + u_j^\star \frac{\partial u_i^\star}{\partial x_j}, v_i \right)_\Omega = 0. \tag{16}$$

2. **Diffusion–pressure step**

$$\begin{aligned} &\rho_f \left(\frac{u_i - u_i^\star}{\Delta t}, v_i \right)_\Omega + \left(\tau_{ij}^f, \frac{\partial v_i}{\partial x_j} \right)_\Omega - \left(p, \frac{\partial v_j}{\partial x_j} \right)_\Omega - \left(\frac{\partial u_j}{\partial x_j}, q \right)_\Omega \\ &\quad + (\rho_s - \rho_f) \left(\frac{u_i - u_i^n}{\Delta t}, v_i \right)_{\Omega_{n+1}^s} + \left(\tau_{ij}^s - \tau_{ij}^f, \frac{\partial v_i}{\partial x_j} \right)_{\Omega_{n+1}^s} \\ &= (\bar{h}_i, v_i)_{\Gamma^N} + \rho_f (g_i, v_i)_\Omega + (\rho_s - \rho_f)(g_i, v_i)_{\Omega_{n+1}^s}. \end{aligned} \tag{17}$$

Here $u_i^\star$ denotes the intermediate velocity obtained from the convection step, while the second stage accounts for viscous, incompressibility, and solid-elastic effects. Both stages are linearized before constructing the algebraic systems.

As in [14], the convection equation is treated using a Taylor–Galerkin approximation, which is standard in finite-element flow computations [16, 17].

The current algorithm avoids a nonlinear solve at each time step through explicit treatment of convection and linearization of the solid terms. A fully nonlinear formulation could instead be treated using Newton or inexact Newton iterations, while retaining the existing distributed data structures and element-wise parallel assembly. This would require the solid configuration, interpolation operator, nonlinear residual, and consistent tangent matrix to be updated at each nonlinear iteration; if convection were treated fully implicitly, the resulting nonsymmetric Jacobian would also favor a Krylov solver such as GMRES rather than PMINRES. Higher-order time integration, for example Crank–Nicolson or BDF2, could likewise be incorporated without changing the basic distributed-memory framework, provided that the inertial, viscous, solid-elastic, and convective terms and the moving-solid update are treated consistently at second order. A semi-implicit formulation that preserves the symmetry of the diffusion–pressure system would be particularly attractive because PMINRES could then be retained. These nonlinear and higher-order extensions, together with their parallel solver performance, will be investigated in future work.

The resulting linearized convection problem is

$$(u_i^\star, v_i)_\Omega = \left(u_i^n - \Delta t\, u_j^n \frac{\partial u_i^n}{\partial x_j}, v_i\right)_\Omega - \frac{\Delta t^2}{2}\left(u_k^n \frac{\partial u_i^n}{\partial x_k}, u_j^n \frac{\partial v_i}{\partial x_j}\right)_\Omega . \tag{18}$$

For the diffusion–pressure step, the linearization described in [14] gives

$$\begin{aligned}
& \rho_f \left(\frac{u_i - u_i^\star}{\Delta t}, v_i\right)_\Omega + (\rho_s - \rho_f)\left(\frac{u_i - u_i^n}{\Delta t}, v_i\right)_{\Omega^s_{n+1}} \\
+ \;& \mu_f \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}, \frac{\partial v_i}{\partial x_j}\right)_\Omega - \left(p, \frac{\partial v_j}{\partial x_j}\right)_\Omega - \left(\frac{\partial u_j}{\partial x_j}, q\right)_\Omega \\
+ \;& \mu_s \Delta t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \Delta t \frac{\partial u_i}{\partial x_k}\frac{\partial u_j^n}{\partial x_k} + \Delta t \frac{\partial u_i^n}{\partial x_k}\frac{\partial u_j}{\partial x_k}, \frac{\partial v_i}{\partial x_j}\right)_{\Omega^s_{n+1}} \\
+ \;& \Delta t^2 \left(\frac{\partial u_i}{\partial x_k}(\tau_{kl}^s)^n \frac{\partial u_j^n}{\partial x_l} + \frac{\partial u_i^n}{\partial x_k}(\tau_{kl}^s)^n \frac{\partial u_j}{\partial x_l}, \frac{\partial v_i}{\partial x_j}\right)_{\Omega^s_{n+1}} \\
+ \;& \Delta t \left(\frac{\partial u_i}{\partial x_k}(\tau_{kj}^s)^n + (\tau_{il}^s)^n \frac{\partial u_j}{\partial x_l}, \frac{\partial v_i}{\partial x_j}\right)_{\Omega^s_{n+1}} \\
= \;& (\bar{h}_i, v_i)_{\Gamma^N} + \rho_f (g_i, v_i)_\Omega + (\rho_s - \rho_f)(g_i, v_i)_{\Omega^s_{n+1}} \\
+ \;& \left(\mu_s \Delta t^2 \frac{\partial u_i^n}{\partial x_k}\frac{\partial u_j^n}{\partial x_k} + \Delta t^2 \frac{\partial u_i^n}{\partial x_k}(\tau_{kl}^s)^n \frac{\partial u_j^n}{\partial x_l} - (\tau_{ij}^s)^n, \frac{\partial v_i}{\partial x_j}\right)_{\Omega^s_{n+1}} .
\end{aligned}$$

For the spatial discretization, we employ a background Eulerian mesh Ω^h for the fixed computational domain and an updated Lagrangian mesh Ω^{sh}_{n+1} for the solid. Following [14], the velocity–pressure approximation on the structured background mesh is the inf–sup stable Taylor–Hood $\mathbf{Q}2$–$\mathbf{Q}1$ pair. In two dimensions, biquadratic interpolation is used for the velocity and bilinear interpolation for the pressure on quadrilateral elements. In three dimensions, the corresponding triquadratic–trilinear $\mathbf{Q}2$–$\mathbf{Q}1$ pair is used on hexahedral elements. Thus, the background velocity is approximated by $\mathbf{Q}2$ functions and the pressure by $\mathbf{Q}1$ functions.

Accordingly, we define

$$V^h(\Omega^h) = \text{span}\{\psi_1, \ldots, \psi_{N^u}\} \subset H^1(\Omega),$$
$$L^h(\Omega^h) = \text{span}\{\phi_1, \ldots, \phi_{N^p}\} \subset L^2(\Omega).$$

The discrete background fields are expanded as

$$\mathbf{u}^h(\mathbf{x}) = \sum_{i=1}^{N^u} \mathbf{u}(\mathbf{x}_i)\psi_i(\mathbf{x}), \qquad p^h(\mathbf{x}) = \sum_{i=1}^{N^p} p(\mathbf{x}_i)\phi_i(\mathbf{x}). \tag{19}$$

The current solid region Ω^{sh}_{n+1} is discretized by $\mathbf{P}1$ simplicial elements, giving the finite-element space

$$V^{sh}(\Omega^{sh}_{n+1}) = \text{span}\{\psi^s_1, \ldots, \psi^s_{N^s}\} \subset H^1(\Omega^s_{n+1}). \tag{20}$$

The background velocity is restricted to the solid mesh through isoparametric interpolation,

$$\mathbf{u}^{sh}(\mathbf{x}) = \sum_{i=1}^{N^s} \mathbf{u}^h(\mathbf{x}^s_i)\psi^s_i(\mathbf{x}) = \sum_{i=1}^{N^s}\sum_{j=1}^{N^u} \mathbf{u}(\mathbf{x}_j)\psi_j(\mathbf{x}^s_i)\psi^s_i(\mathbf{x}), \tag{21}$$

where $\mathbf{x}^s_i$ denotes the current solid nodal coordinates.

It is important to emphasize that no separate no-slip boundary condition is imposed explicitly on the moving fluid–solid interface. The one-field formulation uses a single global velocity field throughout the computational domain, and the velocity at a solid node is obtained by restricting the background velocity to its current position through Eq. (21). The same interpolated velocity is then used to update the Lagrangian solid position. Consequently, the fluid and solid velocities coincide kinematically at the interface by construction. Thus, the no-slip condition is enforced through the common one-field velocity and the interpolation operator P, rather than through an additional interface constraint or independently imposed Dirichlet condition.

Substituting (19), (21), and the corresponding discrete test functions into the linearized variational equations yields the saddle-point system

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}, \tag{22}$$

with

$$\mathbf{A} = \mathbf{M}/\Delta t + \mathbf{K} + \mathbf{D}^T(\mathbf{M}^s/\Delta t + \mathbf{K}^s)\mathbf{D}, \tag{23}$$

and

$$\mathbf{b} = \mathbf{f} + \mathbf{D}^T\mathbf{f}^s + \mathbf{M}\mathbf{u}^\star/\Delta t + \mathbf{D}^T\mathbf{M}^s\mathbf{D}\mathbf{u}^n/\Delta t. \tag{24}$$

Here $\mathbf{D}$ is the interpolation operator that maps background-grid velocities to the solid-grid representation:

$$\mathbf{D} = \begin{bmatrix} \mathbf{P}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^T \end{bmatrix}, \qquad \mathbf{P}_{ij} = \psi_i(\mathbf{x}^s_j). \tag{25}$$

The matrices $\mathbf{M}$ and $\mathbf{K}$ are the mass and stiffness matrices assembled over the Eulerian mesh Ω^h, while $\mathbf{M}^s$ and $\mathbf{K}^s$ denote the corresponding matrices assembled over the solid mesh Ω^{sh}_{n+1}. The vector $\mathbf{f}$ contains the body-force and traction contributions on the background mesh, and $\mathbf{f}^s$ denotes the corresponding contribution from the solid subdomain. Finally, $\mathbf{B}$ and $\mathbf{B}^T$ are the discrete divergence and gradient operators associated with the incompressibility constraint in (19).

4 Numerical implementation

This section describes the distributed-memory realization of the one-field monolithic fictitious-domain method. We summarize the parallel framework, data layout and finite-element assembly, linear solver and preconditioner, and the communication and mesh-coupling details that are important for scalable computation.

4.1 Parallel implementation framework

Our implementation extends the one-field monolithic fictitious-domain method of [14] to distributed-memory computation. At each time step, the discretization produces a large sparse saddle-point system for the global velocity and pressure unknowns. Moderate three-dimensional background meshes, such as $16 \times 16 \times 16$ or $32 \times 32 \times 32$, remain feasible in serial, whereas finer discretizations such as $64 \times 64 \times 64$ become sufficiently expensive to motivate parallel solution.

The implementation has three main components: finite-element discretization on the fixed Eulerian background mesh and moving Lagrangian solid mesh, solid–fluid coupling through the isoparametric interpolation matrix $\mathbf{P}$ in (25), and solution of the resulting linear system. Only the Eulerian background mesh is decomposed among MPI processes; the complete solid mesh is replicated on all processes, as detailed in Section 4.2. Element computations and interpolation contributions associated with a background subdomain are therefore local, with communication required primarily for neighboring guard data and global solver operations. The principal scalability challenge is the iterative linear solution, particularly the loss of preconditioner effectiveness when inter-subdomain couplings are omitted.

4.2 Data structures and computational setup

The code is implemented within the CAMPFIRE framework [18], with the structured background grid managed by PARAMESH [19]. PARAMESH represents the domain as a collection of structured blocks. Cell-centered values are stored in `unk`; in two dimensions, `unk_e_x` and `unk_e_y` store values at horizontal- and vertical-edge midpoints, respectively, and `unk_n` stores corner values. Figure 2 illustrates these locations. In three dimensions, `facevarx`, `facevary`, and `facevarz` store values at the centers of the yz-, zx-, and xy-faces. The discussion below focuses on the two-dimensional layout.

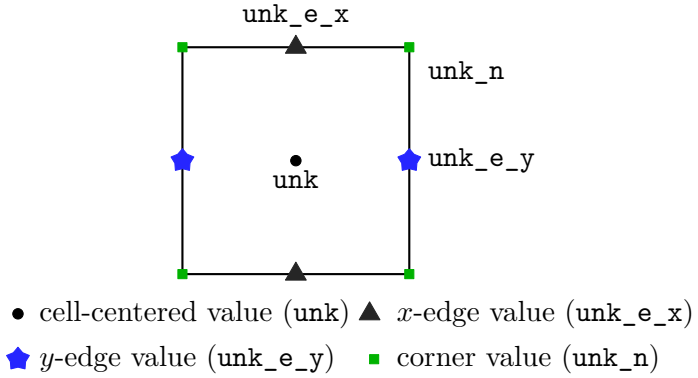


Figure 2: Storage locations of discrete variables on a two-dimensional PARAMESH cell. Cell-centered values are stored in `unk`, values at the centers of the horizontal (x-) edges are stored in `unk_e_x`, values at the centers of the vertical (y-) edges are stored in `unk_e_y`, and corner values are stored in `unk_n`.

Each two-dimensional background cell is treated as a quadrilateral finite element with biquadratic velocity and bilinear pressure interpolation, as described in Section 3. Thus, an

element has 9 degrees of freedom for each velocity component and 4 for the pressure, giving 22 unknowns in total. The arrays `unk`, `unk_e_x`, `unk_e_y`, and `unk_n` together provide the required velocity locations, while the four pressure degrees of freedom are stored at the element corners in `unk_n`.

Only the Eulerian background mesh is decomposed among MPI processes. The complete Lagrangian solid mesh, including its nodal coordinates, connectivity, and geometry, is stored on every process. Consequently, the solid configuration and the matrices $\mathbf{M}^s$ and $\mathbf{K}^s$ are updated and assembled from the full solid mesh on each process, so no structural repartitioning is required during deformation.

The MPI decomposition enters through the solid–fluid coupling. Each process identifies the solid nodes relevant to its local background blocks and evaluates the corresponding interpolation contributions; this local set changes as the solid crosses subdomain boundaries, while the replicated solid mesh remains unchanged.

For a solid node $\boldsymbol{x}_j^s$ associated with a local background block, the containing Eulerian element is located before the corresponding entries of P are formed. Because the background mesh is Cartesian and block structured, the block is identified from its bounding box and the cell indices follow directly from the nodal coordinates and grid spacing. In two dimensions, for a candidate cell $[x_i, x_{i+1}] \times [y_k, y_{k+1}]$,

$$\xi_j = 2\frac{x_j^s - x_i}{x_{i+1} - x_i} - 1, \qquad \eta_j = 2\frac{y_j^s - y_k}{y_{k+1} - y_k} - 1.$$

The node is accepted when

$$-1 - \varepsilon \le \xi_j, \eta_j \le 1 + \varepsilon,$$

where ε accounts for points on element boundaries. The three-dimensional case additionally uses ζ_j. The nonzero interpolation entries are then obtained from the local **Q**2 basis functions,

$$P_{ij} = \psi_i(\boldsymbol{x}_j^s).$$

Point location and interpolation are repeated after each solid-position update.

Each block is surrounded by guard cells and guard nodes that hold neighboring data needed near block interfaces, allowing local element evaluation after guard-cell exchange.

For a 3×3 block in two dimensions (with the third spatial index fixed, say $k = 1$), $\texttt{unk}(1, i, j, 1, lb)$ uses $i, j = 1, \ldots, 4$. The edge arrays $\texttt{unk_e_x}(1, i, j, 1, lb)$ and $\texttt{unk_e_y}(1, i, j, 1, lb)$ use index ranges $(i, j) = (1{:}4, 1{:}5)$ and $(1{:}5, 1{:}4)$, respectively, while $\texttt{unk_n}(1, i, j, 1, lb)$ uses $(1{:}5, 1{:}5)$. Figure 3 illustrates the corresponding active and guard locations for a 2×2 block.

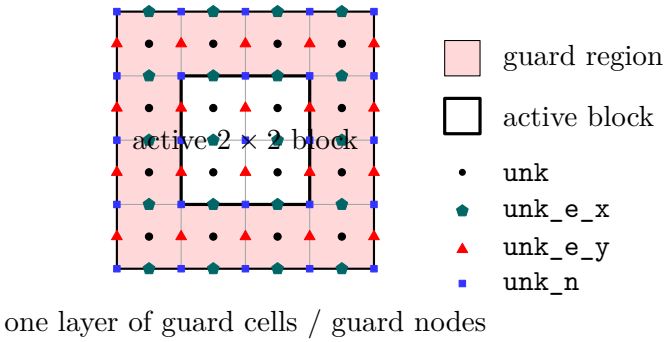


Figure 3: Layout of a 2×2 computational block with one layer of guard cells and guard nodes.

Figure 4 illustrates four neighboring 2×2 blocks. The blue-shaded regions denote guard layers adjacent to the physical boundary, whereas the red-shaded regions contain data received from neighboring blocks.

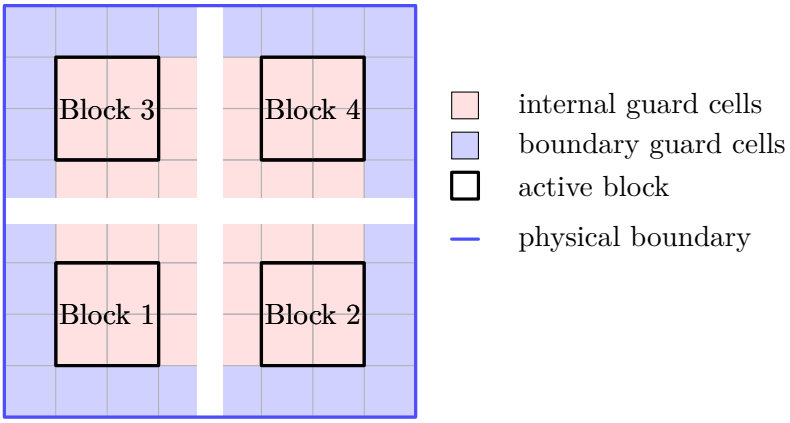


Figure 4: Schematic of four neighboring 2×2 blocks. The red-shaded regions denote internal guard cells obtained from adjacent blocks, whereas the blue-shaded regions indicate guard cells adjacent to the physical boundary.

During assembly, element matrices and vectors are evaluated on both active cells and the required guard cells; see the red-shaded regions in Fig. 4. For the two-dimensional problem, each element contributes a 22×22 matrix. Including guard-cell elements ensures that the equations associated with the active degrees of freedom match those of the sequential assembly without special interface corrections.

Boundary conditions are imposed at the algebraic level. For a Dirichlet degree of freedom, the corresponding row is replaced by a unit diagonal row and the right-hand side is set to the prescribed value. The same treatment is applied consistently on the distributed grid.

4.3 Linear system solver

The coupled system is solved by preconditioned MINRES (PMINRES) with split preconditioning, summarized in Algorithm 1. For an initial guess $\boldsymbol{x}_0$, MINRES is applied to

$$\widetilde{A}(\boldsymbol{q}) = M_1^{-1} A(M_2^{-1}\boldsymbol{q}), \qquad \widetilde{\boldsymbol{b}} = M_1^{-1}\boldsymbol{b}.$$

The iteration is terminated when $\|\boldsymbol{b} - A\boldsymbol{x}\|_2/\|\boldsymbol{b}\|_2$ is below the prescribed tolerance, with the physical solution recovered as

$$\boldsymbol{x} = \boldsymbol{x}_0 + M_2^{-1}\boldsymbol{y}.$$

To accelerate convergence, we use the block preconditioner

$$\mathcal{M} = \begin{bmatrix} \mathbf{A}' & 0 \\ 0 & \mathbf{M}_p/\Delta t \end{bmatrix}, \tag{26}$$

where

$$\mathbf{A}' = \mathbf{M}/\Delta t + \mathbf{K},$$

and $\mathbf{M}_p$ is the pressure mass matrix. An incomplete Cholesky factorization with zero fill-in (IC0) is applied to $\mathcal{M}$; if L denotes its lower triangular factor, the split preconditioner is

$$M_1 = L, \qquad M_2 = L^T.$$

In parallel, off-process couplings are omitted when constructing the factors. After reordering by subdomain, this gives the preconditioner a block-Jacobi character. The resulting triangular solves are local and inexpensive, but the preconditioner is weaker than its serial counterpart.

For multi-block grids, guard values must be synchronized before operator applications that require off-process data. The solver working vectors are stored locally, so a vector requiring communication is copied to the corresponding PARAMESH arrays, updated through `pf_guardcell`,

and then returned to the solver workspace. A mapping from global unknown numbers to `unk`, `unk_e_x`, `unk_e_y`, and `unk_n` is constructed before time stepping and reused throughout the solve. Sparse matrix–vector products and triangular solves use CSR storage within this distributed data layout.

Algorithm 1 Preconditioned MINRES with split preconditioner

Input: symmetric matrix/operator A, right-hand side $\boldsymbol{b}$
tolerance `tol`, maximum number of iterations `maxit`
split SPD preconditioner solves $M_1^{-1}(\cdot)$ and $M_2^{-1}(\cdot)$, initial guess $\boldsymbol{x}_0$

Output: approximate solution $\boldsymbol{x}$

Define
$$\widetilde{A}(\boldsymbol{q}) = M_1^{-1} A\Big(M_2^{-1}\boldsymbol{q}\Big), \qquad \widetilde{\boldsymbol{b}} = M_1^{-1}\boldsymbol{b}$$

$\boldsymbol{x} \leftarrow \boldsymbol{x}_0, \quad \boldsymbol{r}_0 \leftarrow \boldsymbol{b} - A\boldsymbol{x}, \quad \widetilde{\boldsymbol{r}}_0 \leftarrow M_1^{-1}\boldsymbol{r}_0$
$\beta_1 \leftarrow \|\widetilde{\boldsymbol{r}}_0\|_2$
if $\|\boldsymbol{r}_0\|_2/\|\boldsymbol{b}\|_2 \le$ `tol` **then**
  return $\boldsymbol{x}$
end if
$\boldsymbol{y} \leftarrow \boldsymbol{0}, \quad \boldsymbol{v} \leftarrow \widetilde{\boldsymbol{r}}_0/\beta_1, \quad \boldsymbol{v}_0 \leftarrow \boldsymbol{0}$
$\boldsymbol{z}_1 \leftarrow \boldsymbol{0}, \quad \boldsymbol{z}_2 \leftarrow \boldsymbol{0}$
$\mathtt{cs} \leftarrow 1, \quad \mathtt{sn} \leftarrow 0, \quad \bar{\mathrm{d}} \leftarrow 0, \quad \epsilon \leftarrow 0, \quad \bar{\phi} \leftarrow \beta_1$
for $k = 1, 2, \ldots,$ `maxit` **do**
  $\boldsymbol{w} \leftarrow \widetilde{A}(\boldsymbol{v})$
  if $k > 1$ **then**
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \beta_1 \cdot \boldsymbol{v}_0$
  end if
  $\alpha \leftarrow \boldsymbol{v}^T\boldsymbol{w}, \quad \boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \cdot \boldsymbol{v}$
  $\beta \leftarrow \|\boldsymbol{w}\|_2, \quad \boldsymbol{v}_0 \leftarrow \boldsymbol{v}$
  if $\beta > 0$ **then**
    $\boldsymbol{v} \leftarrow \boldsymbol{w}/\beta$
  else
    $\boldsymbol{v} \leftarrow \boldsymbol{0}$
  end if
  $\epsilon_{old} \leftarrow \epsilon$
  $\delta \leftarrow \mathtt{cs} \cdot \bar{\mathrm{d}} + \mathtt{sn} \cdot \alpha, \quad \gamma \leftarrow \mathtt{sn} \cdot \bar{\mathrm{d}} - \mathtt{cs} \cdot \alpha$
  $\epsilon \leftarrow \mathtt{sn} \cdot \beta, \quad \bar{\mathrm{d}} \leftarrow -\mathtt{cs} \cdot \beta$
  $\rho \leftarrow \sqrt{\gamma^2 + \beta^2}$
  if $\rho = 0$ **then**
    break
  end if
  $\mathtt{cs} \leftarrow \gamma/\rho, \quad \mathtt{sn} \leftarrow \beta/\rho$
  $\phi \leftarrow -\mathtt{cs} \cdot \bar{\phi}, \quad \bar{\phi} \leftarrow \mathtt{sn} \cdot \bar{\phi}$
  $\boldsymbol{z} \leftarrow \boldsymbol{v}_0 - \epsilon_{old} \cdot \boldsymbol{z}_1 - \delta \cdot \boldsymbol{z}_2, \quad \boldsymbol{z} \leftarrow \boldsymbol{z}/\rho$
  $\boldsymbol{y} \leftarrow \boldsymbol{y} + \phi \cdot \boldsymbol{z}, \quad \boldsymbol{x} \leftarrow \boldsymbol{x}_0 + M_2^{-1}\boldsymbol{y}$
  $\boldsymbol{z}_1 \leftarrow \boldsymbol{z}_2, \quad \boldsymbol{z}_2 \leftarrow \boldsymbol{z}$
  if $\|\boldsymbol{b} - A\boldsymbol{x}\|_2/\|\boldsymbol{b}\|_2 \le$ `tol` **then**
    break
  end if
  if $\beta = 0$ **then**
    break
  end if
  $\beta_1 \leftarrow \beta$
end for
return $\boldsymbol{x}$

4.4 Technical considerations

Two implementation issues deserve additional attention. First, the mapping between global algebraic unknowns and the PARAMESH storage arrays must remain consistent during assembly, operator application, residual evaluation, and guard-cell communication. The mapping introduced above is therefore constructed once and reused, avoiding repeated index searches during PMINRES. Second, the block-Jacobi approximation trades preconditioner strength for locality: omitting off-process couplings reduces communication in the triangular solves but can increase the iteration count. The numerical experiments in Section 5 quantify this trade-off and its effect on strong scaling.

The relative fluid and solid mesh resolutions must also be controlled. Following Wang et al. [14], the local ratio of background- to solid-element measure is kept within a suitable range, using areas in two dimensions and volumes in three dimensions. To allow for subsequent deformation, the initial diameter of a solid element is also restricted to approximately two local Cartesian grid spacings. In parallel, two guard-cell layers are used so that solid elements spanning an internal subdomain boundary can be represented and assembled consistently, as illustrated in Figure 5.

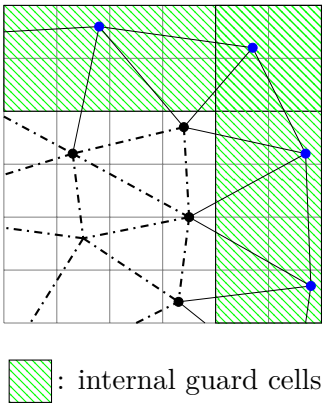


Figure 5: Illustration of solid elements and nodes near an internal subdomain boundary. Two layers of guard cells are required so that element contributions involving nodes across the boundary can be assembled consistently.

5 Numerical experiments

In this section, we assess the proposed parallel algorithm using four benchmark problems: a deformable disc and a stiffer deformable slab in two-dimensional lid-driven cavities, followed by deformable sphere and cylinder problems in three dimensions. The examples are designed to provide complementary assessments of numerical consistency, discretization sensitivity, solver robustness, and parallel scalability. Depending on the benchmark, we report serial–parallel comparisons, temporal and spatial refinement results, physically relevant solution measures, wall-clock times, speedups, parallel efficiencies, PMINRES convergence behavior, and representative solution fields.

For the two-dimensional examples, we consider runs with up to 256 MPI ranks. For the three-dimensional examples, we test scalability up to 512 MPI ranks for the sphere case and up to 256 MPI ranks for the cylinder case. In all experiments, one MPI rank is assigned to one CPU core. All computations were performed on the Taiwania 3 supercomputer at the National Center for High-performance Computing (NCHC), Taiwan. Taiwania 3 is a CPU-based system with 900 compute nodes and 50,400 CPU cores, connected by an InfiniBand HDR100 high-speed

interconnect, with a reported peak performance of approximately 2.7 PFLOPS.

To support long three-dimensional simulations, we implemented a checkpoint–restart mechanism that saves the current state before the wall-time limit is reached and resumes the computation in a subsequent job. This makes it possible to carry out long transient simulations reliably across multiple job submissions.

In the following subsections, we describe the geometry and physical parameters of each benchmark and then examine the corresponding parallel performance.

5.1 Deformable disc in lid-driven cavity flow

We first consider a standard benchmark consisting of a deformable disc immersed in a two-dimensional lid-driven cavity flow; see Fig. 6. The fluid domain is the unit square

$$\Omega = (0,1) \times (0,1).$$

The top boundary moves horizontally with velocity $(u, v) = (1, 0)$, whereas the left, right, and bottom boundaries are stationary and satisfy the no-slip condition. To remove the pressure indeterminacy, we prescribe $p = 0$ at one corner of the cavity.

The immersed solid is initially a stress-free circular disc of radius $r = 0.2$ centered at $(0.6, 0.5)$. The fluid and solid densities are both set to one, the fluid viscosity is $\mu_f = 0.01$, and the solid is modeled as an incompressible neo-Hookean material with shear modulus $G = \mu_s = 0.1$. The geometric and physical parameters are summarized in Table 1.

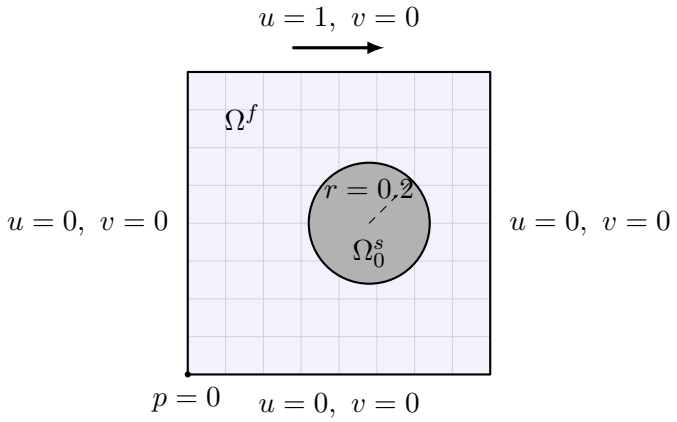


Figure 6: Deformable disc in a lid-driven cavity.

L	H	r	(x_c, y_c)	(ρ_f, μ_f)	(ρ_s, G)
1.0	1.0	0.2	$(0.6, 0.5)$	(1, 0.01)	(1, 0.1)

Table 1: Parameters for the deformable-disc test.

For the main simulation, the fluid domain is discretized using a 256×256 quadrilateral mesh, while the solid disc is represented by 31,163 triangular elements with 15,794 nodes. The time-step size is $\Delta t = 0.0025$, and the complete simulation is advanced to $T = 8$, corresponding to 3200 time steps. The serial and parallel solutions are compared at $T = 1$ and $T = 2$. For the temporal-refinement study, the time-step sizes $\Delta t = 0.008$, 0.004, 0.002, and 0.001 are considered while the fluid and solid meshes are held fixed. For the background-mesh refinement study, $\Delta t = 0.001$ is fixed and the Eulerian meshes 32^2, 64^2, 128^2, and 256^2 are used with the same solid mesh. Parallel performance is evaluated from the first 1000 time steps, as reported in Table 6, together with the corresponding average PMINRES iteration counts.

We first examine the consistency between the serial and parallel implementations. For the solid coordinates, the root-mean-square (RMS) difference is defined as

$$E^s_{\mathrm{RMS}} = \left[\frac{1}{N_s} \sum_{i=1}^{N_s} \left\{ (x_i^P - x_i^S)^2 + (y_i^P - y_i^S)^2 \right\} \right]^{1/2}, \tag{27}$$

where S and P denote the serial and parallel solutions, respectively.

Table 2: RMS differences in the solid coordinates between the serial and parallel computations. The serial solution is used as the reference and $\Delta t = 0.0025$.

	Solid-coordinate RMS difference, E^s_{RMS}	
n_p	$T = 1$	$T = 2$
4	1.641×10^{-7}	2.689×10^{-7}
16	2.254×10^{-7}	5.276×10^{-7}
64	2.355×10^{-7}	5.218×10^{-7}
256	2.524×10^{-7}	6.062×10^{-7}

For $q \in \{u, v, p\}$, the RMS difference in the corresponding fluid variable is defined by

$$E_{\mathrm{RMS}}(q) = \left[\frac{1}{N_q} \sum_{i=1}^{N_q} \left(q_i^P - q_i^S \right)^2 \right]^{1/2}, \tag{28}$$

where N_q is the number of degrees of freedom associated with q.

Table 3: RMS differences in the velocity and pressure fields between the serial and parallel computations at $T = 1$ and $T = 2$. The serial solution is used as the reference.

	$T = 1$			$T = 2$		
n_p	$E_{\mathrm{RMS}}(u)$	$E_{\mathrm{RMS}}(v)$	$E_{\mathrm{RMS}}(p)$	$E_{\mathrm{RMS}}(u)$	$E_{\mathrm{RMS}}(v)$	$E_{\mathrm{RMS}}(p)$
4	3.342×10^{-8}	2.852×10^{-8}	3.736×10^{-7}	7.229×10^{-8}	6.282×10^{-8}	8.016×10^{-7}
16	4.623×10^{-8}	4.351×10^{-8}	5.437×10^{-7}	1.038×10^{-7}	1.115×10^{-7}	1.276×10^{-6}
64	6.704×10^{-8}	6.252×10^{-8}	7.736×10^{-7}	1.633×10^{-7}	1.441×10^{-7}	1.693×10^{-6}
256	8.752×10^{-8}	6.941×10^{-8}	8.900×10^{-7}	2.804×10^{-7}	2.041×10^{-7}	3.862×10^{-6}

Tables 2 and 3 demonstrate close agreement between the serial and parallel solutions. The RMS differences in the solid coordinates remain below 6.1×10^{-7} at both times. The velocity differences are of order 10^{-8}–10^{-7}, while the pressure differences remain below 3.9×10^{-6}. Although the differences increase slightly at $T = 2$, they remain small for all processor counts and exhibit no systematic deterioration as n_p increases. These results confirm that the parallel implementation reproduces the serial discrete solution with high accuracy.

The temporal- and background-mesh refinement studies are carried out using 256 MPI processes to reduce the computational cost. The physical parameters and solver tolerances are kept fixed, while either the time-step size or the Eulerian mesh size is varied. The preceding serial–parallel comparisons provide confidence that the observed changes under refinement are associated primarily with the numerical discretization rather than the parallel implementation.

For temporal refinement, the fluid and solid meshes are held fixed and the time-step size is successively reduced. The RMS difference between two successive solutions is evaluated as

$$E_{\mathrm{RMS}}(q) = \| q_{\Delta t} - q_{\Delta t/2} \|_{\mathrm{RMS}}.$$

Table 4: Temporal convergence for the deformable-disc benchmark at $T = 1$. The computations use 256 MPI processes. For each row, $E_{\mathrm{RMS}}(q)$ denotes the RMS difference between the solution computed with time step Δt and that computed with the refined time step $\Delta t/2$.

Δt	$E_{\mathrm{RMS}}(u)$	Rate	$E_{\mathrm{RMS}}(v)$	Rate	$E_{\mathrm{RMS}}(p)$	Rate
0.008	1.427×10^{-4}	–	9.501×10^{-5}	–	1.172×10^{-4}	–
0.004	7.154×10^{-5}	0.996	4.766×10^{-5}	0.995	5.947×10^{-5}	0.979
0.002	3.581×10^{-5}	0.999	2.385×10^{-5}	0.999	3.367×10^{-5}	0.821

Table 4 shows a consistent reduction in the RMS differences as the time-step size is decreased. For the finest pair, the $\Delta t = 0.002$ solution is compared with the $\Delta t = 0.001$ solution. The observed rates for both velocity components are essentially one over the refinement sequence, consistent with the first-order temporal discretization. The pressure follows the same general trend, with rates close to one on the coarser refinement levels and a moderately reduced rate for the finest pair.

We next examine sensitivity to the Eulerian background mesh. To isolate the effect of the fluid discretization, the solid triangulation is kept fixed while the background fluid grid is successively refined. The errors are evaluated between consecutive mesh levels at coincident Eulerian nodes.

Table 5: Background-mesh refinement for the deformable-disc benchmark. The solid mesh is kept fixed, and the errors are computed between successive fluid-grid refinements at coincident nodes.

Mesh pair	$E_{\mathrm{RMS}}(u)$	Rate	$E_{\mathrm{RMS}}(v)$	Rate
32^2–64^2	1.330×10^{-2}	–	7.699×10^{-3}	–
64^2–128^2	2.637×10^{-3}	2.334	2.469×10^{-3}	1.641
128^2–256^2	1.519×10^{-3}	0.796	1.346×10^{-3}	0.875

Table 5 shows that the velocity differences decrease monotonically as the Eulerian background mesh is refined while the solid mesh is held fixed. The combined velocity difference decreases from 1.537×10^{-2} for the 32^2–64^2 pair to 3.613×10^{-3} for the 64^2–128^2 pair and 2.030×10^{-3} for the 128^2–256^2 pair. An observed rate of approximately two is obtained over the coarser refinement levels, whereas the rate decreases to about 0.83 for the finest pair. The reduction in the observed rate is reasonable because the solid discretization is held fixed and, as the Eulerian mesh becomes finer, the effects of the fixed solid resolution, fluid–solid coupling, temporal discretization, and the corner singularities associated with the lid-driven boundary condition become relatively more significant. Accordingly, this study is interpreted as demonstrating convergence with respect to the Eulerian background mesh rather than the formal asymptotic order of the Q2 velocity approximation.

We next assess the parallel performance and identify the principal sources of computational cost. For this benchmark, the most significant change in the iterative solver occurs when moving from the serial computation to the domain-decomposed parallel computation. The average PMINRES iteration count increases from 588 in serial to 993 on 4 MPI ranks and then rises only gradually to 1044 on 256 MPI ranks. Thus, most of the degradation in preconditioner effectiveness occurs when the global preconditioner is replaced by its block-Jacobi-type parallel approximation; further subdivision of the domain has a comparatively smaller influence on the iteration count. Despite the increased number of iterations, the wall-clock time decreases from 11,025 minutes in serial to 210 minutes on 256 MPI ranks, corresponding to a speedup of 52.51.

The parallel efficiency decreases to 20.5% at the largest processor count, reflecting the increasing communication and synchronization overhead.

First 1000 time steps	**Serial**	**4**	**16**	**64**	**256**
Wall-clock time (minutes)	11025	5081	907	531	210
Speedup	1.00	2.17	12.16	20.78	52.51
Parallel efficiency	100%	54.3%	76.0%	32.5%	20.5%
Average PMINRES iterations	588	993	1005	1020	1044

Table 6: Parallel performance for the deformable-disc case.

To examine the scaling behavior in more detail, Table 7 separates the matrix assembly and linear system solution stages for one representative time step. The matrix assembly stage scales comparatively well, retaining 43.2% efficiency on 256 MPI ranks. The linear solver stage is less scalable, with its efficiency decreasing to 15.4%. This difference indicates that the principal limitation arises from the iterative solution of the coupled saddle-point system rather than from element-level assembly. The slightly superlinear assembly efficiency observed for 4 MPI ranks is plausibly associated with improved cache utilization and reduced memory pressure in the distributed computation.

The computation-to-communication ratio per PMINRES iteration varies between 1.16 and 2.29. The most favorable ratio occurs on 16 MPI ranks, where local computational work and communication are most effectively balanced. At larger processor counts, the local subproblems become smaller, while guard-cell exchanges and global reductions account for a larger fraction of each iteration. Consequently, the linear solver loses parallel efficiency more rapidly than the matrix assembly stage.

One time step	**Serial**	**4**	**16**	**64**	**256**
Matrix assembly stage time (seconds)	78.5	19.08	5.49	1.78	0.71
Speedup	1.00	4.11	14.30	44.10	110.56
Parallel efficiency	100%	102.8%	89.4%	68.9%	43.2%
Linear system solution stage time (seconds)	547.0	221.9	45.47	20.65	13.90
Speedup	1.00	2.46	12.03	26.49	39.35
Parallel efficiency	100%	61.6%	75.2%	41.4%	15.4%
Computation/communication ratio per PMINRES iteration	–	1.16	2.29	1.84	1.50

Table 7: Stage timings for the deformable-disc case.

To provide quantitative measures of the solid motion in addition to the flow-field visualizations, Table 8 reports the area-weighted centroid of the disc together with the centroid, RMS, and maximum nodal displacements relative to the initial configuration. The centroid initially moves predominantly toward the left, from $(0.6000, 0.5000)$ at $T = 0$ to $(0.4088, 0.4963)$ at $T = 2$, following the primary lid-driven circulation. At later times, the motion develops a substantial upward component, with the centroid reaching $(0.2358, 0.7819)$ at $T = 4$. Accordingly, the centroid displacement increases from 8.271×10^{-3} at $T = 0.25$ to 4.605×10^{-1} at $T = 4$. The RMS and maximum nodal displacements exhibit the same increasing trend, reaching 4.750×10^{-1} and 5.876×10^{-1}, respectively, at $T = 4$. These quantitative measures complement the velocity and pressure snapshots and demonstrate the substantial translation and deformation experienced by the immersed solid under the lid-driven flow.

Table 8: Evolution of the centroid and displacement measures of the deformable disc.

T	x_c	y_c	d_c	d_{RMS}	$d_{\max}$
0.25	0.592032	0.497780	8.271×10^{-3}	8.946×10^{-3}	1.399×10^{-2}
0.50	0.576760	0.493904	2.403×10^{-2}	2.580×10^{-2}	3.872×10^{-2}
1.00	0.531849	0.485925	6.959×10^{-2}	7.289×10^{-2}	9.798×10^{-2}
2.00	0.408785	0.496274	1.913×10^{-1}	1.943×10^{-1}	2.132×10^{-1}
3.00	0.283383	0.594337	3.304×10^{-1}	3.351×10^{-1}	3.646×10^{-1}
4.00	0.235808	0.781866	4.605×10^{-1}	4.750×10^{-1}	5.876×10^{-1}

Finally, Figs. 7 and 8 illustrate the evolution of the deformable disc and the surrounding flow. The computed motion agrees qualitatively with the results reported in [14, 20]. Driven by the primary clockwise circulation generated by the moving lid, the initially circular disc is transported through the cavity and undergoes substantial time-dependent deformation. As the disc approaches the upper wall, it is stretched and flattened by the strong shear near the moving lid, while narrow fluid gaps remain between the solid and the cavity boundaries. It subsequently moves away from the upper wall and returns toward a more compact configuration as the coupled motion continues.

The velocity snapshots show that the dominant lid-driven recirculation is maintained throughout the simulation, while the moving and deforming disc locally modifies the surrounding flow. The velocity vectors bend smoothly around the solid, with strong gradients developing near the disc and within the narrow fluid gaps. The pressure distribution also changes continuously with the disc motion, reflecting the evolving hydrodynamic loading on the solid. Localized pressure extrema occur near the upper corners of the cavity, where the moving-lid boundary condition meets the stationary side walls. These corner effects also explain why the global pressure field is particularly sensitive to background-mesh refinement. No visible discontinuities are observed at subdomain interfaces in either the velocity or pressure fields, providing further qualitative evidence of consistency in the parallel fluid–structure interaction computation.

5.2 Deformation of a slab in lid-driven cavity flow

We next consider a more demanding lid-driven cavity problem involving a slender and substantially stiffer immersed structure. The solid is a thin rectangular slab, initially attached to the bottom wall and centered horizontally in the cavity; see Fig. 9. The fluid domain is again

$$\Omega = (0, 1) \times (0, 1).$$

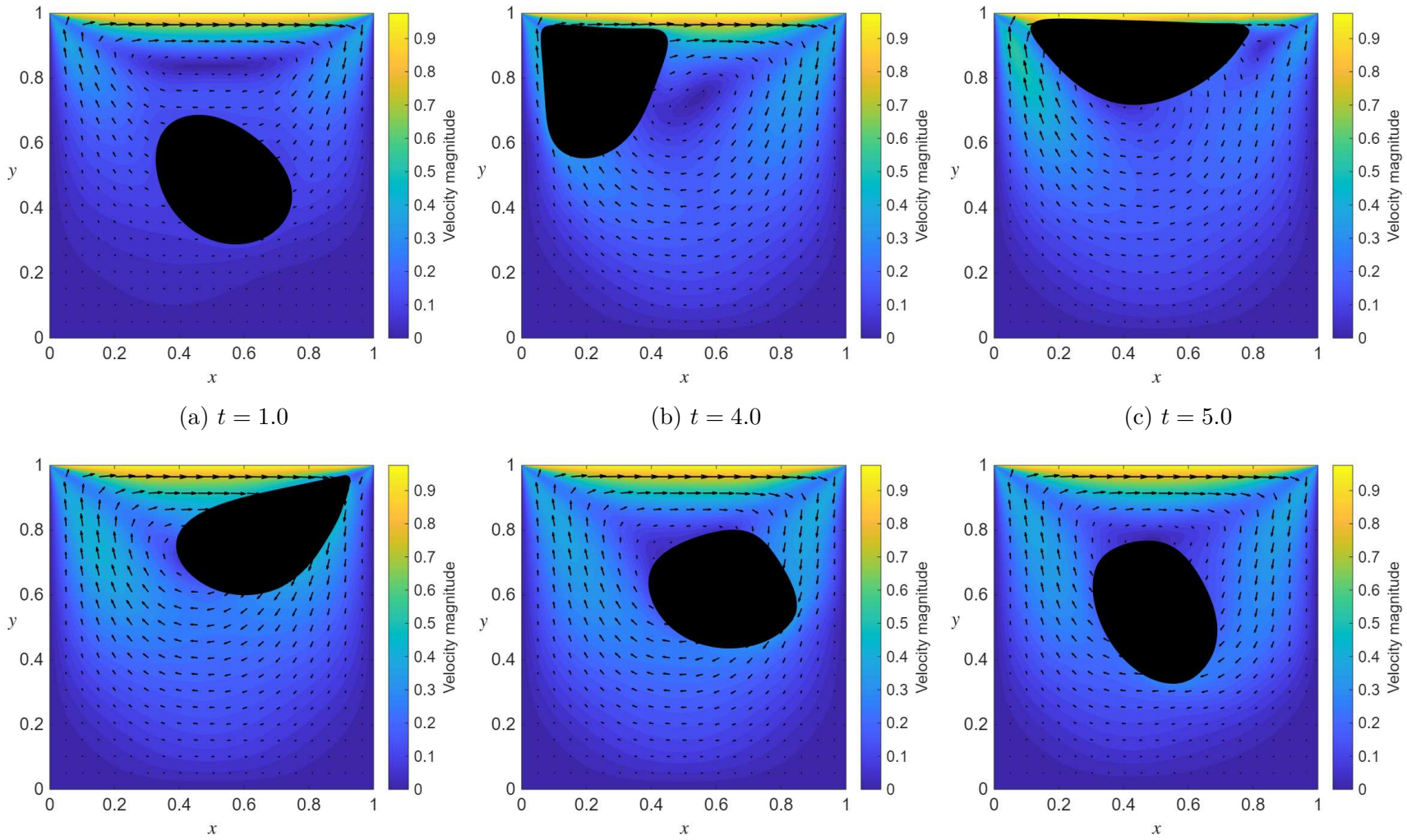
The top wall moves with velocity $(u, v) = (1, 0)$, while the remaining boundaries satisfy the no-slip condition. As in the deformable-disc benchmark, the pressure is fixed at one corner point.

The slab has width $w = 0.05$ and height $h = 0.7$. The fluid domain is discretized using a 160×160 quadrilateral mesh, while the slab is represented by 4000 linear triangular elements with 2111 nodes. The time step is $\Delta t = 0.001$, and the simulation is advanced to $t = 4$. All physical parameters are the same as those used for the deformable disc, except that the solid shear modulus is increased to

$$G = \mu_s = 10,$$

making the structure substantially stiffer. The geometric and physical parameters are summarized in Table 9.

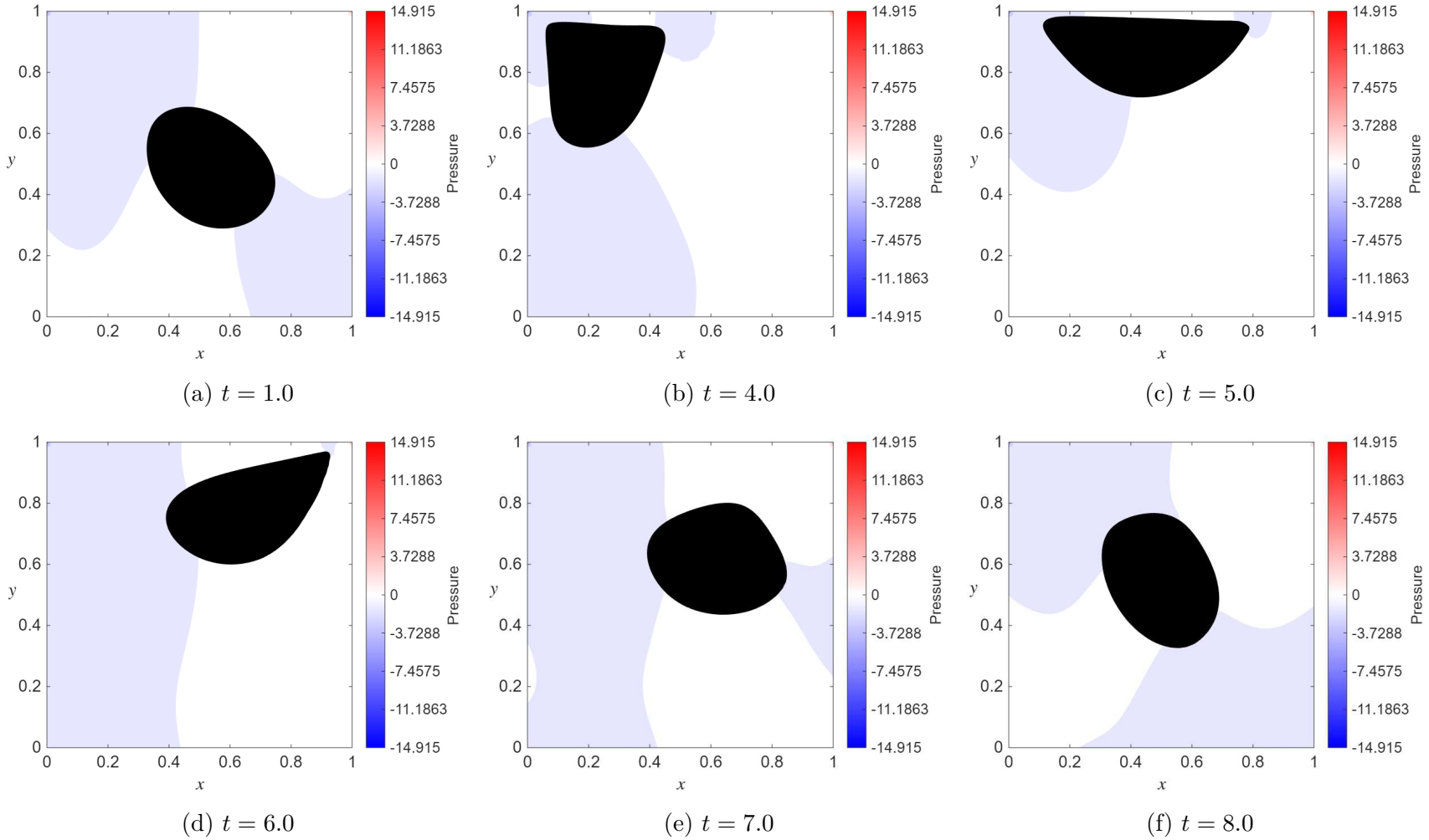
We first assess the consistency of the parallel implementation by comparing the parallel solutions with the corresponding serial solution. Table 10 reports the RMS differences in the solid coordinates at $T = 1$, using the serial computation as the reference.



(a) $t = 1.0$ (b) $t = 4.0$ (c) $t = 5.0$

(d) $t = 6.0$ (e) $t = 7.0$ (f) $t = 8.0$

Figure 7: Evolution of the velocity field and deformable disc in the lid-driven cavity at six representative times. The contours show the velocity magnitude, the arrows indicate the velocity direction, and the deformed disc is shown in black. A common color scale is used in all panels.



(a) $t = 1.0$ (b) $t = 4.0$ (c) $t = 5.0$

(d) $t = 6.0$ (e) $t = 7.0$ (f) $t = 8.0$

Figure 8: Evolution of the pressure field and deformable disc in the lid-driven cavity at six representative times. The deformed disc is shown in black. A common symmetric color scale centered at zero is used in all panels.

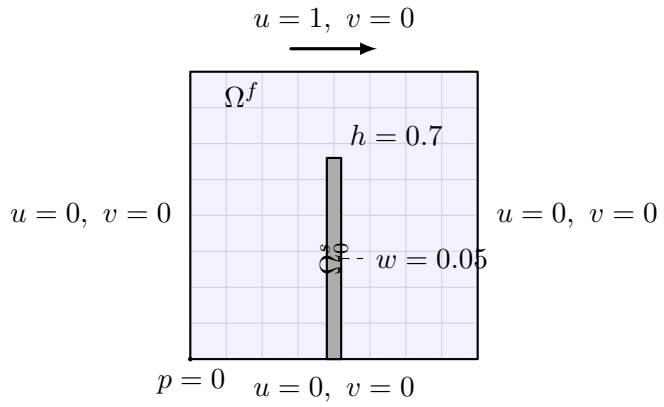


Figure 9: Deformable slab in a lid-driven cavity.

Parameter	**Value**	**Parameter**	**Value**
Fluid domain	$L = 1.0,\ H = 1.0$	Slab width	$w = 0.05$
Background mesh	160×160	Slab height	$h = 0.7$
Fluid density	$\rho_f = 1$	Solid density	$\rho_s = 1$
Fluid viscosity	$\mu_f = 0.01$	Shear modulus	$G = \mu_s = 10$
Solid mesh	4000 triangular elements	Solid nodes	2111

Table 9: Parameters for the deformable-slab test.

The RMS differences remain of order 10^{-4} for all processor counts, indicating close agreement between the parallel and serial solutions, although a slight increase is observed as the number of MPI processes increases. These differences are larger than those obtained for the deformable-disc benchmark, reflecting the greater numerical difficulty of the slab problem. As shown by the analysis of the linear system solution stage below, PMINRES reaches the prescribed maximum of 5000 iterations at some time steps in all parallel runs. This behavior is associated with the more challenging saddle-point systems arising from the slender and relatively stiff solid, together with the sensitivity of the block-Jacobi-type preconditioner to the domain decomposition. Consequently, part of the observed serial–parallel discrepancy is likely due to incomplete convergence of the iterative solver at those time steps rather than to the parallel discretization itself.

We next examine the parallel performance of the longer simulations to $T = 4$, using 1, 4, 16, 64, and 256 MPI ranks. The results over the first 2000 time steps are summarized in Table 11. In contrast to the deformable-disc benchmark, the average PMINRES iteration count varies nonmonotonically with the number of MPI ranks. It increases from 2044 in the serial

Table 10: RMS differences in the solid coordinates between the serial and parallel solutions for the deformable-slab benchmark at $T = 1$. The serial solution is used as the reference.

n_p	E^s_{RMS}
4	7.940×10^{-4}
16	8.040×10^{-4}
64	8.107×10^{-4}
256	8.335×10^{-4}

computation to 2173 on 4 MPI ranks, reaches 3744 on 16 MPI ranks, decreases to 2326 on 64 MPI ranks, and then increases again to 2567 on 256 MPI ranks. This behavior indicates that the performance of the block-Jacobi-type preconditioner depends not only on the number of subdomains but also on how the domain decomposition interacts with the slender immersed structure.

The variation in solver convergence is reflected directly in the parallel performance. The 16-rank case has the largest average PMINRES iteration count and consequently achieves a comparatively low parallel efficiency of 51.4%. By contrast, the 64-rank case requires fewer iterations and reaches a substantially larger speedup of 34.93. At some time steps, PMINRES reaches the maximum allowed number of iterations, namely 5000, demonstrating that the stiff and slender slab generates more difficult coupled linear systems than the deformable-disc benchmark. Nevertheless, the wall-clock time decreases from 9816 minutes in serial to 154 minutes on 256 MPI ranks, corresponding to a speedup of 63.74.

First 2000 time steps	**Serial**	**4**	**16**	**64**	**256**
Wall-clock time (minutes)	9816	2660	1194	281	154
Speedup	1.00	3.69	8.22	34.93	63.74
Parallel efficiency	100%	92.2%	51.4%	54.6%	24.9%
Average PMINRES iterations	2044	2173	3744	2326	2567

Table 11: Parallel performance for the deformable-slab case.

To identify the principal sources of computational cost, Table 12 separates the matrix assembly and linear system solution stages for one representative time step. The matrix assembly stage exhibits comparatively regular scaling and reaches a speedup of 80.00 on 256 MPI ranks. The linear solver stage is less regular because it is directly affected by the nonmonotone PMINRES iteration counts. In particular, although the computation-to-communication ratio on 16 MPI ranks is relatively favorable, the much larger iteration count at that processor count limits the reduction in total solver time.

The computation-to-communication ratio remains between 1.73 and 2.10 over the tested parallel runs. Thus, communication overhead is important but does not by itself explain the observed scaling behavior. The more significant factor is the interaction among the slender solid geometry, the subdomain partitioning, and the resulting block-Jacobi-type preconditioner. The slab benchmark therefore provides a useful test of both parallel scalability and linear-solver robustness for a geometry-sensitive FSI configuration.

One time step	**Serial**	**4**	**16**	**64**	**256**
Matrix assembly stage time (seconds)	8.8	2.3	0.65	0.24	0.11
Speedup	1.00	3.82	13.53	36.67	80.00
Parallel efficiency	100%	95.5%	84.6%	57.3%	31.3%
Linear system solution stage time (seconds)	255.7	86.65	35.57	8.2	4.5
Speedup	1.00	2.95	7.18	29.96	54.60
Parallel efficiency	100%	73.75%	44.8%	46.8%	21.3%
Computation/communication ratio per PMINRES iteration	–	1.78	2.10	1.94	1.73

Table 12: Stage timings for the deformable-slab case.

Finally, Figs. 10 and 11 illustrate the coupled evolution of the slab and the surrounding flow.

As the lid-driven circulation develops, the initially vertical slab bends progressively under the hydrodynamic loading, with its free end displaced toward the left while the lower end remains attached to the bottom wall. The deformation continues throughout the simulated interval, while its rate of change becomes more gradual at later times.

The velocity snapshots show that the primary lid-driven recirculation is maintained while the deforming slab produces a strong local modification of the flow. The velocity vectors vary smoothly around the structure, with pronounced gradients developing in its vicinity. The pressure distribution changes consistently with the slab deformation, and the resulting pressure loading, together with the viscous stresses, contributes to the bending of the structure. Both the velocity and pressure fields remain smooth across the computational domain, with no visible discontinuities at subdomain interfaces. Taken together with the serial–parallel comparison and the performance results above, these observations show that the parallel implementation can accommodate a relatively stiff and slender immersed structure, while also highlighting the increased difficulty of the associated iterative linear solves.

5.3 Motion of a sphere in 3D lid-driven cavity flow

As a three-dimensional counterpart of the deformable-disc benchmark, we next consider the motion and deformation of a sphere in a three-dimensional lid-driven cavity. The fluid occupies the unit cube

$$\Omega = (0,1)^3.$$

The initial solid is a stress-free sphere of radius $r = 0.2$, centered at $(0.6, 0.5, 0.5)$; see Fig. 12. The top lid moves with velocity $(u, v, w) = (1, 0, 0)$, while the remaining walls are fixed and satisfy the no-slip condition. The pressure is pinned at one corner point.

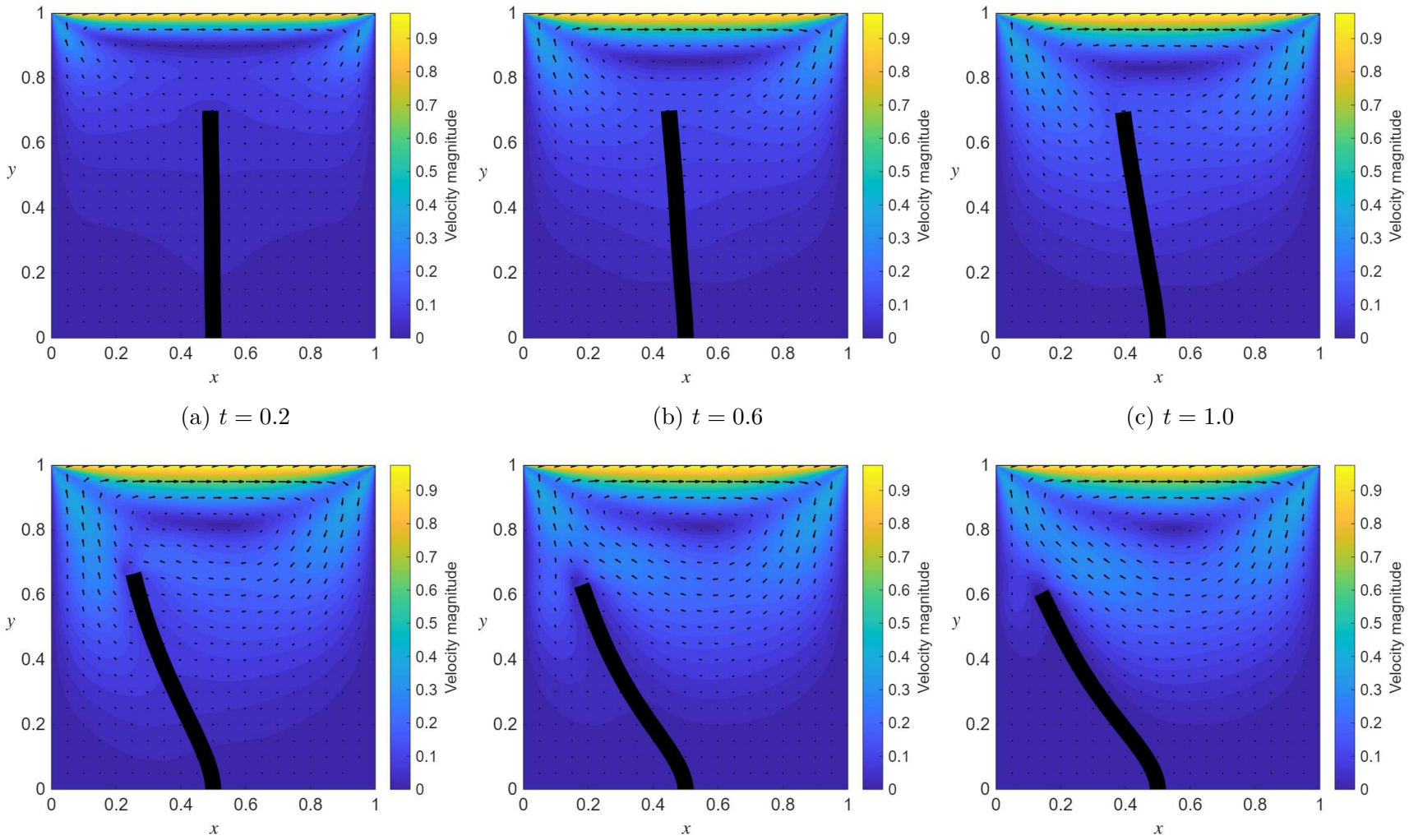
The physical parameters are chosen to match those of the two-dimensional deformable-disc benchmark: $\rho_f = \rho_s = 1$, $\mu_f = 0.01$, and $G = \mu_s = 0.1$. For the main three-dimensional simulation, the background fluid mesh consists of $64 \times 64 \times 64$ cells, while the sphere is discretized by 68,280 tetrahedral elements with 12,478 nodes. The time-step size is $\Delta t = 0.0025$, and the simulation is advanced to $T = 8$. Parallel performance is assessed using the wall-clock time over the first 500 time steps. The parameters are summarized in Table 13.

Parameter	**Value**	**Parameter**	**Value**
Domain	$(0,1)^3$	Sphere radius	$r = 0.2$
Sphere center	$(0.6, 0.5, 0.5)$	Fluid mesh	$64 \times 64 \times 64$
Solid mesh	68280 tetrahedra	Solid nodes	12478
Fluid density	$\rho_f = 1$	Solid density	$\rho_s = 1$
Fluid viscosity	$\mu_f = 0.01$	Shear modulus	$G = \mu_s = 0.1$
Time step	$\Delta t = 0.0025$	Final time	$T = 8$

Table 13: Parameters for the 3D sphere test.

To verify that the parallel consistency observed in the two-dimensional benchmarks also extends to three dimensions, an additional computation is performed on a 32^3 background mesh using several domain decompositions. This smaller mesh is used only for the consistency study so that a serial reference computation remains practical. All computations use the same time-step size $\Delta t = 0.0025$ and are advanced to $T = 1$. Table 14 reports the RMS differences between the serial solution and the parallel solutions obtained using $n_p = 8$, 64, and 512 MPI processes.

For all three parallel runs, the RMS differences in the individual velocity components remain of order 10^{-8}–10^{-7}, while the pressure differences remain below 5.4×10^{-7}. The combined velocity difference increases from 8.045×10^{-8} for $n_p = 8$ to 1.150×10^{-7} for $n_p = 64$ and



(a) $t = 0.2$ (b) $t = 0.6$ (c) $t = 1.0$

(d) $t = 2.0$ (e) $t = 3.0$ (f) $t = 4.0$

Figure 10: Evolution of the velocity field and deformable slab in the lid-driven cavity at six representative times. The contours show the velocity magnitude, the arrows indicate the velocity direction, and the deformed slab is shown in black. A common color scale is used in all panels.

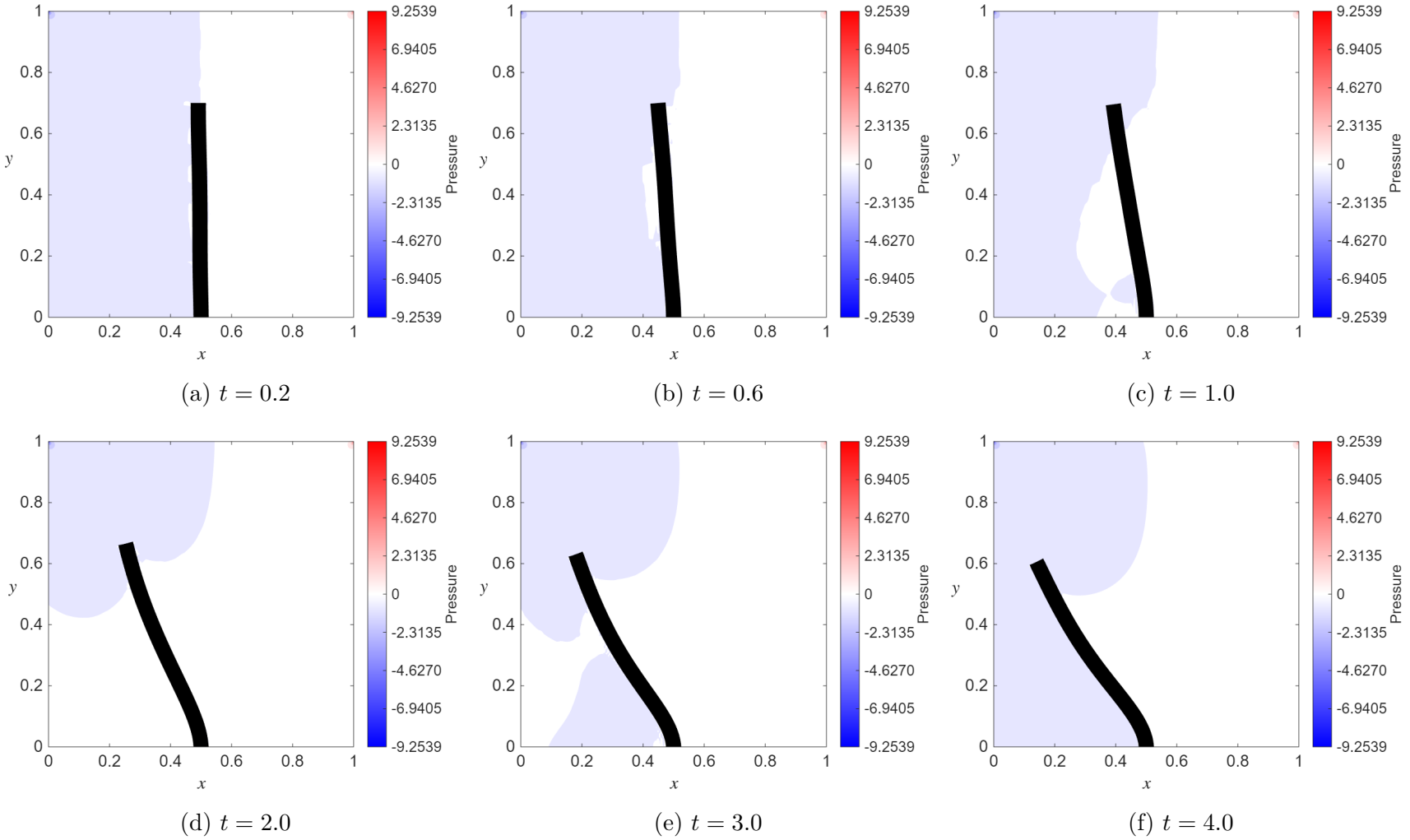


Figure 11: Evolution of the pressure field and deformable slab in the lid-driven cavity at six representative times. The deformed slab is shown in black. A common symmetric color scale centered at zero is used in all panels.

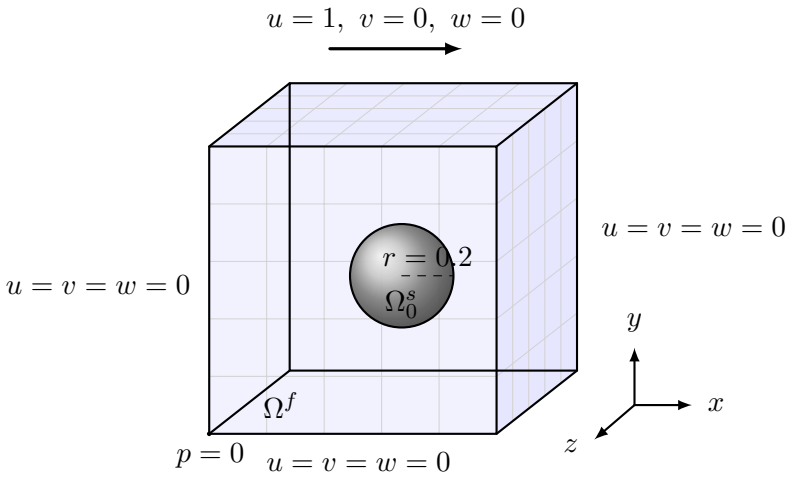


Figure 12: Deformable sphere in a three-dimensional lid-driven cavity.

1.653×10^{-7} for $n_p = 512$. Although the discrepancies increase slightly as the number of MPI subdomains increases, they remain very small over the entire range of decompositions. These results confirm that, at $T = 1$, the three-dimensional parallel implementation reproduces the corresponding serial discrete solution with high accuracy.

Table 14: Parallel consistency for the three-dimensional deformable-sphere benchmark at $T = 1$ on the 32^3 background mesh using $\Delta t = 0.0025$. The serial solution is used as the reference, and the RMS differences are evaluated at the common global Q2 and Q1 nodes.

n_p	$E_{\mathrm{RMS}}(u)$	$E_{\mathrm{RMS}}(v)$	$E_{\mathrm{RMS}}(w)$	$E_{\mathrm{RMS}}(\boldsymbol{u})$	$E_{\mathrm{RMS}}(p)$
8	4.513×10^{-8}	2.775×10^{-8}	4.294×10^{-8}	8.045×10^{-8}	2.242×10^{-7}
64	7.727×10^{-8}	4.258×10^{-8}	7.377×10^{-8}	1.150×10^{-7}	2.903×10^{-7}
512	1.249×10^{-7}	5.147×10^{-8}	9.522×10^{-8}	1.653×10^{-7}	5.303×10^{-7}

We next return to the main 64^3 simulation, which uses the same time-step size $\Delta t = 0.0025$ but is advanced to $T = 8$. The parallel performance of this substantially larger problem is evaluated over the first 500 time steps, as reported in Table 15. This three-dimensional problem is considerably more demanding than the two-dimensional benchmarks. With the prescribed PMINRES tolerance of 10^{-8}, the solver often reaches the imposed maximum of 1000 iterations after the initial time steps, while the relative residual typically remains between 10^{-6} and 10^{-7}. The example is therefore used primarily to assess large-scale parallel computability and solver behavior in a challenging three-dimensional regime rather than to demonstrate convergence of the iterative solver to the prescribed tolerance at every time step.

Despite this more demanding solver behavior, the parallel gain is substantial. The wall-clock time decreases from 56,500 minutes in serial to 470 minutes on 512 MPI ranks, corresponding to a speedup of 120.2. The parallel efficiency decreases as the number of ranks increases, reaching 23% on 512 MPI ranks, but the resulting acceleration is nevertheless sufficient to make this large-scale three-dimensional monolithic FSI computation substantially more practical.

To examine the scaling behavior in greater detail, Table 16 separates the matrix assembly and linear system solution stages for one representative time step. The matrix assembly stage exhibits strong scaling, achieving a speedup of 311.8 and a parallel efficiency of 60.9% on 512 MPI ranks. In contrast, the linear solver stage scales less effectively, with an efficiency of only 14.5% at the same processor count. Because PMINRES reaches the prescribed maximum number of iterations in this example, the solver time reflects both the increased difficulty of the three-

First 500 time steps	Serial	8	64	512
Wall-clock time (minutes)	56500	9791	2096	470
Speedup	1.00	5.77	26.95	120.2
Parallel efficiency	100%	72%	42%	23%

Table 15: Parallel performance for the 3D sphere case.

dimensional saddle-point system and the growing communication cost of the domain-decomposed computation.

The effect of communication is also evident from the computation-to-communication ratio per PMINRES iteration. This ratio decreases from 3.66 on 8 MPI ranks to 1.73 on 64 MPI ranks and 0.87 on 512 MPI ranks. Thus, at the largest processor count, the communication time per iteration exceeds the local computational work. This trend is more pronounced than in the two-dimensional benchmarks and identifies interprocessor communication, together with the limited effectiveness of the current preconditioner, as the principal factors restricting the scalability of the three-dimensional linear solver.

One time step	Serial	8	64	512
Matrix assembly stage time (seconds)	2501	304.2	46.2	8.02
Speedup	1.00	8.22	54.13	311.8
Parallel efficiency	100%	102.8%	84.6%	60.9%
Linear system solution stage time (seconds)	3948.8	866.1	214.8	53.1
Speedup	1.00	4.56	18.38	74.36
Parallel efficiency	100%	57.0%	28.7%	14.5%
Computation/communication ratio per PMINRES iteration	–	3.66	1.73	0.87

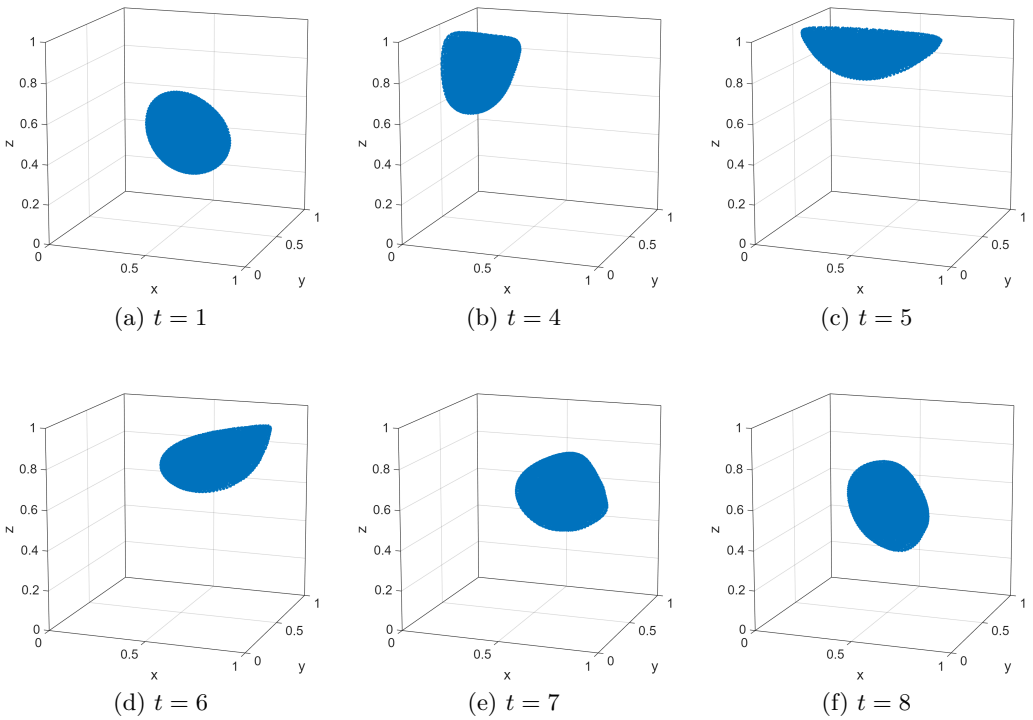
Table 16: Stage timings for the 3D sphere case.

Figure 13 shows representative snapshots of the three-dimensional solid motion. Driven by the lid-induced circulation, the initially spherical body is transported through the cavity and undergoes substantial deformation. The evolution is qualitatively consistent with that observed for the two-dimensional deformable disc, while the three-dimensional geometry permits additional deformation in the transverse direction. The sequence of configurations remains smooth throughout the simulated interval, demonstrating that the method can track large three-dimensional solid motion on a fixed Eulerian background mesh.

Although the prescribed PMINRES tolerance is not reached at every time step in the main 64^3 simulation, the independent 32^3 serial–parallel comparison demonstrates close agreement among different domain decompositions, while the 64^3 results demonstrate substantial parallel acceleration for a much larger problem. Taken together, these results show that the proposed implementation extends effectively to three-dimensional FSI computations, while also identifying the iterative solver and communication overhead as the principal targets for further improvement.

5.4 Deformation of a cylinder in a lid-driven cavity flow

We next consider a three-dimensional lid-driven cavity problem containing a deformable cylinder, which serves as the three-dimensional counterpart of the slab example. The fluid domain is the



(a) $t = 1$ (b) $t = 4$ (c) $t = 5$

(d) $t = 6$ (e) $t = 7$ (f) $t = 8$

Figure 13: Snapshots of the deformable sphere in the three-dimensional lid-driven cavity at selected times.

rectangular cavity

$$\Omega = (0, L) \times (0, W) \times (0, H), \qquad L = 1,\ W = 0.5,\ H = 1.$$

A vertical cylinder of radius $r = 0.05$ is initially attached to the bottom wall, with its axis parallel to the z-direction and its base centered at $(0.5, 0.25, 0)$; see Fig. 14. The initial cylinder height is $h_c = 0.7$, so that the cylinder occupies $0 \le z \le h_c$. As in the previous lid-driven cavity benchmarks, at time $t = 0$ the top lid begins to move uniformly in the positive x-direction with velocity $(u, v, w) = (1, 0, 0)$, while all other walls remain fixed and satisfy the no-slip condition. The pressure is pinned at one corner point in order to remove the constant-pressure indeterminacy.

This benchmark assesses the behavior of the monolithic parallel solver for a relatively slender immersed solid in three dimensions. Compared with the spherical case, the cylinder undergoes a more pronounced bending-dominated deformation and therefore provides a more demanding test of the coupled fluid–structure formulation. The fluid and solid densities are both taken to be one, the fluid viscosity is set to $\mu_f = 0.01$, and the solid is modeled as an incompressible neo-Hookean material with shear modulus

$$G = \mu_s = 10.$$

The geometric and material parameters are summarized in Table 17.

The background fluid mesh is discretized by $48 \times 24 \times 48$ hexahedral cells, while the cylinder is represented by a tetrahedral mesh containing 6043 elements and 1387 nodes. The simulation uses a time step $\Delta t = 5 \times 10^{-4}$ and is advanced for 6000 time steps. These discretization and time-stepping parameters are listed in Table 18.

The timing results for the first 1000 time steps are listed in Table 19. The wall-clock time decreases from 8220 minutes in serial to 3520, 1696, and 301 minutes on 4, 32, and 256 MPI ranks, respectively. These timings correspond to speedups of 2.33, 4.84, and 27.3. The speedup is

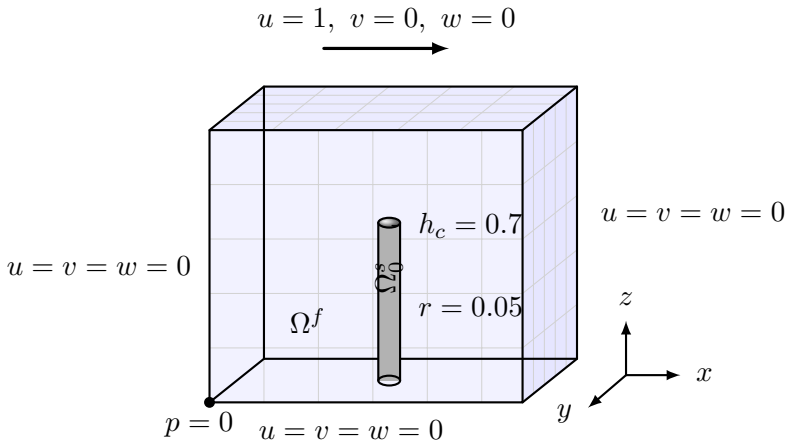


Figure 14: Deformable cylinder in a three-dimensional lid-driven cavity.

Fluid	**Solid cylinder**
$L = 1.0$	$r = 0.05,\ h_c = 0.7$
$W = 0.5$	base center $(x_c, y_c, z_c) = (0.5, 0.25, 0)$
$H = 1.0$	axis parallel to the z-direction
$\rho_f = 1$	$\rho_s = 1$
$\mu_f = 0.01$	$G = \mu_s = 10$

Table 17: Parameters for the 3D cylinder test.

significant in absolute terms, but the parallel efficiency is lower than in the sphere case, especially at 32 and 256 MPI ranks.

The lower efficiency is closely related to the increase in PMINRES iteration counts under domain decomposition. The average iteration count increases from 48 in serial to 106, 180, and 201 on 4, 32, and 256 MPI ranks, respectively. Compared with the sphere, the cylinder is a more slender structure and its deformation is closer to a bending-dominated cantilever response. The resulting coupled linear systems are more sensitive to the domain decomposition, and the block-Jacobi-type preconditioner loses effectiveness more rapidly. Nevertheless, the 256-rank computation still reduces the total runtime by more than a factor of 27, making this three-dimensional cylinder simulation substantially more affordable than the serial computation.

Table 20 further separates the matrix assembly and linear solver stages for one representative time step. The matrix assembly stage still shows useful parallel scaling, with a speedup of 87.75 on 256 MPI ranks and an efficiency of 34.3%. The solver stage is much less favorable: on 4 MPI ranks, the solver time is slightly larger than in the serial run, and on 256 MPI ranks the solver efficiency is only 3.6%. This indicates that the cost reduction from distributing the local algebraic work is partly offset by the weaker preconditioner and by communication overhead.

The communication bottleneck is particularly clear in the computation-to-communication ratio per PMINRES iteration. This ratio decreases from 2.30 on 4 MPI ranks to 0.33 on 256 MPI ranks. Hence, at the largest core count, communication dominates the local computational work within each PMINRES iteration. Together with the increasing iteration count, this explains why the cylinder benchmark has the lowest solver-stage efficiency among the tests considered here. The case is therefore a useful stress test for future improvements in scalable preconditioning for slender three-dimensional immersed structures.

Figure 15 displays the deformation history of the cylinder at several representative time instants. At an early stage, the cylinder remains nearly vertical and only a small displacement

Fluid discretization	Solid discretization / time stepping
background mesh: $48 \times 24 \times 48$	6043 tetrahedral elements
hexahedral cells	1387 nodes
	$\Delta t = 5 \times 10^{-4}$
	6000 time steps

Table 18: Discretization and time stepping for the 3D cylinder test.

First 1000 time steps	Serial	4	32	256
Wall-clock time (minutes)	8220	3520	1696	301
Speedup	1.00	2.33	4.84	27.3
Parallel efficiency	100%	58%	15%	11%
Average PMINRES iterations	48	106	180	201

Table 19: Parallel performance for the 3D cylinder case.

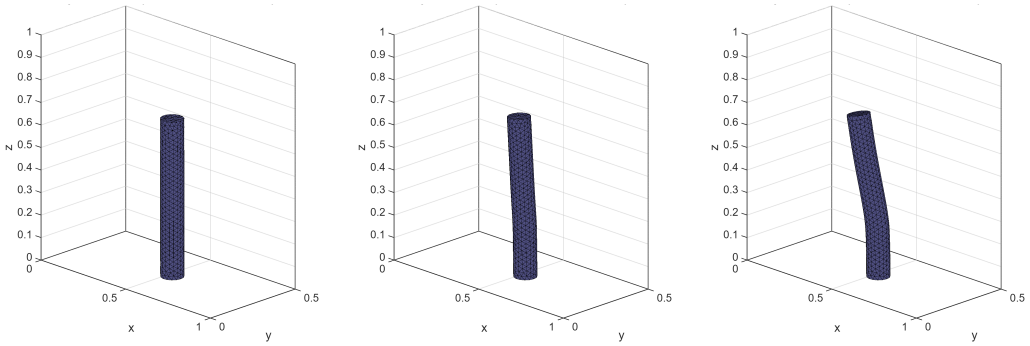
is visible. As time increases, the lid-driven recirculating flow exerts a progressively stronger drag on the upper portion of the cylinder, while the bottom end remains attached to the cavity floor. As a result, the cylinder develops a clear bending deformation, with the displacement increasing from the base toward the top. This is characteristic of a cantilever-type response. From $t = 0.05$ to $t = 0.5$, the deformation is still relatively mild, but by $t = 1$ and $t = 1.5$ the upper part of the cylinder has already bent noticeably toward the negative x-direction. At later times, the cylinder continues to lean and its centerline becomes more curved, while the deformation evolves more slowly. This suggests that the coupled fluid–structure system is approaching a nearly steady deformed configuration. The dominant motion is bending rather than compression or torsion, which is consistent with the slender geometry of the immersed solid and the horizontal shear generated by the moving lid.

6 Conclusion

We have developed a distributed-memory parallel implementation of a monolithic one-field fictitious-domain finite-element method for incompressible fluid–structure interaction. The coupled saddle-point systems are solved by PMINRES with a symmetric split preconditioner

One time step	Serial	4	32	256
Matrix assembly stage time (seconds)	261.5	72.5	12.1	2.98
Speedup	1.00	3.61	21.6	87.75
Parallel efficiency	100%	90.0%	67.5%	34.3%
Linear system solution stage time (seconds)	187.6	223.05	70.1	20.4
Speedup	1.00	0.84	2.67	9.19
Parallel efficiency	100%	21.0%	8.4%	3.6%
Computation/communication ratio per PMINRES iteration	–	2.30	2.20	0.33

Table 20: Stage timings for the 3D cylinder case.



(a) $t = 0.05$ (b) $t = 0.5$ (c) $t = 1$

(d) $t = 1.5$ (e) $t = 2$ (f) $t = 3$

Figure 15: Snapshots of the deformable cylinder in the three-dimensional lid-driven cavity at selected times.

based on a block-diagonal approximation. In parallel, omission of off-process couplings gives the preconditioner a block-Jacobi character, providing a simple and practical baseline for distributed-memory computation.

The numerical experiments demonstrate the applicability of the framework to a range of two- and three-dimensional FSI problems involving soft and stiff immersed bodies with both compact and slender geometries. The two-dimensional disc and slab benchmarks achieve substantial speedups on up to 256 MPI ranks, while the three-dimensional sphere and cylinder examples demonstrate the feasibility of significantly larger simulations for which serial computation becomes prohibitively expensive. Serial–parallel comparisons, temporal and background-mesh refinement studies, and quantitative solid-motion measures further support the numerical consistency of the implementation and the physical plausibility of the computed solutions.

Across the benchmarks, finite-element assembly scales more favorably than the linear system solution. The principal limitation on parallel efficiency is the deterioration of the iterative solver under domain decomposition. As the number of MPI subdomains increases, omission of inter-subdomain couplings weakens the block-Jacobi-type preconditioner, leading to larger PMINRES iteration counts together with increased communication and synchronization costs. This effect is particularly pronounced for three-dimensional problems and for slender or relatively stiff immersed structures. Nevertheless, substantial reductions in wall-clock time are obtained even when the strong-scaling efficiency is modest, making otherwise very expensive FSI simulations computationally practical.

The present implementation also has several limitations. The accuracy of the fictitious-domain coupling depends on the relative resolutions of the Eulerian and Lagrangian meshes, and the background-mesh study in Section 5.1 shows that substantially coarser fluid grids do not generally reproduce the resolution obtained on finer meshes. In addition, the first-order temporal discretization limits the achievable time accuracy. These observations motivate both higher-order time integration and stronger parallel preconditioners that retain more inter-subdomain coupling information.

Several extensions are therefore of interest. One possibility is the three-step fractional-step scheme proposed by Wang et al. [21], in which each time step is separated into convection, diffusion, and pressure subproblems. The diffusion stage can be treated efficiently using preconditioned CG, while the pressure stage leads to a comparatively benign degenerate Stokes problem. Such a splitting strategy may be advantageous for demanding FSI configurations, particularly oscillating or slender structures for which the present monolithic PMINRES solver becomes difficult to scale. A complementary direction is to develop stronger preconditioners for the monolithic formulation that retain more global coupling information and thereby reduce iteration counts and improve strong-scaling performance.

The fictitious-domain framework also has potential for inverse and optimal-control problems. Kurahashi [22] demonstrated that a fictitious-domain finite-element method can be combined with an adjoint-variable formulation for control problems involving a moving body. A similar approach could be developed for the present FSI setting using objective functionals based on measured velocities, pressures, interface displacements, or structural responses. The distributed assembly and domain-decomposition infrastructure developed here would remain useful for the corresponding forward and adjoint computations, although additional challenges would arise from differentiation of the moving solid-to-fluid interpolation operator, treatment of the time-dependent solid configuration in the adjoint equations, and storage or reconstruction of forward states. These topics provide natural directions for extending the present framework beyond forward large-scale FSI simulation.

Author Contributions

Meng-Huo Chen: Conceptualization; Methodology; Software; Validation; Formal analysis; Investigation; Data curation; Visualization; Writing – original draft; Writing – review and editing.

Yongxing Wang: Methodology; Formal analysis; derivation of the finite-element weak formulation; Writing – review and editing.

Peter K. Jimack: Conceptualization; initiation of the general research idea; Writing – review and editing.

Acknowledgments

This research was partially supported by the National Science and Technology Council, R.O.C, through grant No. NSTC 114-2115-M-194-006. The authors also express their sincere thanks to the National Center for High-performance Computing (NCHC) for providing the computational and storage resources used in this work.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] L. Formaggia, A. Quarteroni, and A. Veneziani (eds.), *Cardiovascular Mathematics: Modeling and Simulation of the Circulatory System*, Springer, Milan, 2009. https://doi.org/10.1007/978-88-470-1152-6.

[2] M. Heil, *An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems*, Computer Methods in Applied Mechanics and Engineering, 193(1–2) (2004), pp. 1–23. https://doi.org/10.1016/j.cma.2003.09.006.

[3] E. H. Dowell and K. C. Hall, *Modeling of fluid–structure interaction*, Annual Review of Fluid Mechanics, 33 (2001), pp. 445–490. https://doi.org/10.1146/annurev.fluid.33.1.445.

[4] T. Sarpkaya, *A critical review of the intrinsic nature of vortex-induced vibrations*, Journal of Fluids and Structures, 19 (2004), pp. 389–447. https://doi.org/10.1016/j.jfluidstructs.2004.02.005.

[5] T. M. Squires and S. R. Quake, *Microfluidics: fluid physics at the nanoliter scale*, Reviews of Modern Physics, 77 (2005), pp. 977–1026. https://doi.org/10.1103/RevModPhys.77.977.

[6] U. Küttler and W. A. Wall, *Fixed-point fluid–structure interaction solvers with dynamic relaxation*, Computational Mechanics, 43(1) (2008), pp. 61–72. https://doi.org/10.1007/s00466-008-0255-5.

[7] J. Degroote, K.-J. Bathe, and J. Vierendeels, *Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction*, Computers & Structures, 87(11–12) (2009), pp. 793–801. https://doi.org/10.1016/j.compstruc.2008.11.013.

[8] M. Heil, A. L. Hazel, and J. Boyle, *Solvers for large-displacement fluid–structure interaction problems: segregated versus monolithic approaches*, Computational Mechanics, 43(1) (2008), pp. 91–101. https://doi.org/10.1007/s00466-008-0270-6.

[9] R. L. Muddle, M. Mihajlović, and M. Heil, *An efficient preconditioner for monolithically-coupled large-displacement fluid–structure interaction problems with pseudo-solid mesh updates*, Journal of Computational Physics, 231(21) (2012), pp. 7315–7334. https://doi.org/10.1016/j.jcp.2012.07.001.

[10] G. Hou, J. Wang, and A. Layton, *Numerical methods for fluid–structure interaction: a review*, Communications in Computational Physics, 12(2) (2012), pp. 337–377. https://doi.org/10.4208/cicp.291210.290411s.

[11] C. Michler, S. J. Hulshoff, E. H. van Brummelen, and R. de Borst, *A monolithic approach to fluid–structure interaction*, Computers & Fluids, 33(5–6) (2004), pp. 839–848. https://doi.org/10.1016/j.compfluid.2003.06.006.

[12] C. S. Peskin, *The immersed boundary method*, Acta Numerica, 11 (2002), pp. 479–517. https://doi.org/10.1017/S0962492902000077.

[13] F. P. T. Baaijens, *A fictitious domain/mortar element method for fluid–structure interaction*, International Journal for Numerical Methods in Fluids, 35(7) (2001), pp. 743–761. https://doi.org/10.1002/1097-0363(20010415)35:7<743::AID-FLD109>3.0.CO;2-A.

[14] Y. Wang, P. K. Jimack, and M. A. Walkley, *A one-field monolithic fictitious domain method for fluid–structure interactions*, Computer Methods in Applied Mechanics and Engineering, 317 (2017), pp. 1146–1168. https://doi.org/10.1016/j.cma.2017.01.023.

[15] D. Jodlbauer, U. Langer, and T. Wick, *Parallel block-preconditioned monolithic solvers for fluid–structure interaction problems*, International Journal for Numerical Methods in Engineering, 117(6) (2019), pp. 623–643. https://doi.org/10.1002/nme.5970.

[16] O. C. Zienkiewicz, R. L. Taylor, and P. Nithiarasu, *The Finite Element Method for Fluid Dynamics*, 6th ed., Elsevier, Oxford, 2005.

[17] J. Donea and A. Huerta, *Finite Element Methods for Flow Problems*, John Wiley & Sons, Chichester, 2003. https://doi.org/10.1002/0470013826.

[18] F. W. Yang, C. E. Goodyer, M. E. Hubbard, and P. K. Jimack, *An optimally efficient technique for the solution of systems of nonlinear parabolic partial differential equations*, Advances in Engineering Software, 103 (2017), pp. 65–84. https://doi.org/10.1016/j.advengsoft.2016.06.003.

[19] P. MacNeice, K. M. Olson, C. Mobarry, R. de Fainchtein, and C. Packer, *PARAMESH: a parallel adaptive mesh refinement community toolkit*, Computer Physics Communications, 126(3) (2000), pp. 330–354. https://doi.org/10.1016/S0010-4655(99)00501-9.

[20] H. Zhao, J. B. Freund, and R. D. Moser, *A fixed-mesh method for incompressible flow–structure systems with finite solid deformations*, Journal of Computational Physics, 227(6) (2008), pp. 3114–3140. https://doi.org/10.1016/j.jcp.2007.11.019.

[21] Y. Wang, P. K. Jimack, and M. A. Walkley, *A theoretical and numerical investigation of a family of immersed finite element methods*, Journal of Fluids and Structures, 91 (2019), 102754. https://doi.org/10.1016/j.jfluidstructs.2019.102754.

[22] T. Kurahashi, *Investigations on boundary temperature control analysis considering a moving body based on the adjoint variable and the fictitious domain finite element methods*, International Journal for Numerical Methods in Engineering, 103(8) (2015), pp. 582–597. https://doi.org/10.1002/nme.4908.