Journal of Computational Physics 454 (2022) 110897

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A strongly coupled immersed boundary method for fluid-structure interaction that mimics the efficiency of stationary body methods

Nirmal J. Nair *, Andres Goza

Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA

ARTICLE INFO

Article history:
Received 10 March 2021
Received in revised form 5 November 2021
Accepted 10 December 2021
Available online 20 January 2022

Keywords:
Immersed boundary
Fluid-structure interaction
Strongly coupled
Non-stationary bodies
Parallel IB

ABSTRACT

Strongly coupled immersed boundary (IB) methods solve the nonlinear fluid and structural equations of motion simultaneously for strongly enforcing the no-slip constraint on the body. Handling this constraint requires solving several large dimensional systems that scale by the number of grid points in the flow domain even though the nonlinear constraints scale only by the small number of points used to represent the fluid-structure interface. These costly large scale operations for determining only a small number of unknowns at the interface creates a bottleneck to efficiently time-advancing strongly coupled IB methods. In this manuscript, we present a remedy for this bottleneck that is motivated by the efficient strategy employed in stationary-body IB methods while preserving the favorable stability properties of strongly coupled algorithms—we precompute a matrix that encapsulates the large dimensional system so that the prohibitive large scale operations need not be performed at every time step. This precomputation process yields a modified system of small-dimensional constraint equations that is solved at minimal computational cost while time advancing the equations. We also present a parallel implementation that scales favorably across multiple processors. The accuracy, computational efficiency and scalability of our approach are demonstrated on several two dimensional flow problems. Although the demonstration problems consist of a combination of rigid and torsionally mounted bodies, the formulation is derived in a more general setting involving an arbitrary number of rigid, torsionally mounted, and continuously deformable bodies.

1. Introduction

Immersed boundary (IB) methods are numerical techniques for simulating the flow around bodies. In this framework, the bodies described by Lagrangian points are immersed into the fluid domain discretized by non-body-conforming Eulerian points. The interaction between the fluid and body is achieved via interpolation, which allows for the no-slip condition on the immersed body to be enforced by localized momentum forcing near the body. For flows past bodies that are stationary or undergoing prescribed kinematics, the interpolation operators relating the fluid and structure can be formulated to be independent of time. In this setting, the stresses on the immersed surface that enforce the no-slip constraint can be efficiently obtained via small-dimensional, time-constant linear systems with matrices that can be precomputed before advancing the

* Corresponding author.
E-mail address: njn2@illinois.edu (N.J. Nair).

https://doi.org/10.1016/j.jcp.2021.110897
0021-9991/

equations in time [1–3]. However, in fully coupled fluid-structure interaction (FSI) problems, the unknown structural motion leads to a nonlinear algebraic constraint with time-varying operators that can no longer be efficiently precomputed [4,5].

There are a number of ways to handle this nonlinear constraint arising from the fluid-structure coupling. Weakly coupled IB methods treat the body forces or no-slip constraint explicitly in time. Although this approach removes the need to iterate on a nonlinear system of equations to advance the system in time [6,7], the explicit treatment can impose severe time step restrictions if the structure undergoes large deformations or if the structure-to-fluid mass ratio is low [8–10].

By contrast, strongly coupled IB methods treat the body forces and no-slip constraints implicitly in time, allowing for stable simulations of FSI systems with modest time step sizes. The implicit treatment in these strongly coupled methods necessitates that the fluid and structural equations as well as the nonlinear interface constraint be solved simultaneously via an iterative scheme [11–13]. These iterative approaches require, for each FSI iteration, the solution of a large system of equations involving not only the nonlinear constraint and but also the structural and flow equations that scale with the large number of points in the flow domain. These additional large linear solves, which are not present in the stationary-body setting, arise because of the small-dimensional, time-dependent no-slip condition that scales with the number of points on the fluid-structure interface. The small-dimensional nature of this FSI coupling offers a tantalizing question: can the additional expense, compared with stationary-body problems, of time-advancing fully coupled FSI systems be restricted to small-dimensional systems that scale with the number of points at the fluid-structure interface where the FSI coupling occurs?

Towards this aim, some IB methods have reformulated the fully coupled system of equations via block Gauss-Seidel [14] or block-LU factorization [15], so that the iterations are restricted only to the variables existing on the fluid-structure interface. Yet, a key bottleneck to cost reductions in these reformulations is that there is inevitably a large linear system—that scales with the large number of unknowns in the entire flow domain—that gets embedded within the small dimensional nonlinear FSI coupling equation.

A similar embedding of a large linear system within a small-dimensional matrix is also observed in some non-iterative IB methods [16]. These methods utilize a semi-explicit treatment of the body forces or the no-slip constraint, with the benefit that the system may be advanced in time without iteration. Moreover, these approaches have been demonstrated to have favorable stability properties compared with weakly coupled methods, and are therefore often also referred to as strongly coupled methods. However, in the current work we refer to these methods as semi-strongly coupled because they do not strictly enforce the nonlinear algebraic constraint at a given time step, and often result in a reduction in the temporal accuracy of the solver to first order.[1]

In this article, we focus on these strongly and semi-strongly coupled methods that contain an embedded large system within the small nonlinear FSI coupling equation because of their favorable stability properties and potential for computational efficiency. We note that the embedded large-dimensional solve provides a significant obstacle to any practical benefits associated with the nominally small-dimensional nature of the algebraic systems to be iterated on: merely constructing the small-dimensional matrix is computationally expensive since it entails several linear solves involving the large embedded system. Furthermore, this small coupling matrix is dependent on the time-varying position of the immersed body, and therefore it must be constructed at least once per time step (semi-strongly coupled methods) or once per FSI iteration (strongly-coupled methods). The process of constructing the small-dimensional matrix therefore dominates the computational cost of time-advancing these IB methods. We emphasize that this costly process is in contrast to that for flows past stationary bodies, where the small dimensional coupling matrix is not time dependent. This time independence allows one to precompute the coupling matrix once at the beginning of a simulation, allowing for the full system to be advanced without the bottleneck described above [2,20].

We present an efficient remedy for addressing the embedded large linear solve, towards realizing an iterative time advancement scheme that makes use of the small dimensional nature of the FSI coupling. The proposed approach preserves the favorable stability properties of these strongly and semi-strongly coupled schemes, while mimicking desirable features of the stationary body setting – namely, precomputing a matrix that encapsulates the large linear system so that the several prohibitive large linear solves need not be performed at every time step. We also describe a parallel implementation of our FSI algorithm and demonstrate favorable strong scaling on a relatively large two-dimensional problem. Our formulation is developed for FSI problems involving an arbitrary number of rigid, torsionally mounted, and elastically deformable bodies, though for simplicity of presentation our results focus on a combination of rigid and torsionally mounted bodies.

The remainder of the paper is organized as follows. In Sec. 2, we give a background of the IB method of Goza and Colonius [15], which serves as the basis for the specific algorithm proposed in this article. We emphasize that the proposed approach for efficiently addressing the embedded large linear solve arising from many FSI systems has applicability beyond Goza and Colonius [15]. To demonstrate this fact, we further describe in Sec. 2 how the aforementioned bottleneck appears in a number of semi-strongly coupled and strongly coupled methods. The proposed efficient treatment of the FSI coupling is detailed in Sec. 3, and the strategies for parallel implementation on multiple processors are discussed in Sec. 4. We demonstrate the accuracy, computational efficiency and scalability of our approach on several two-dimensional (2D) flow problems in Sec. 5. Finally, conclusions are offered in Sec. 6.

[1] We note that some semi-strongly coupled IB methods [17–19] do not have an embedding of the large system due to their specific formulations. However, these methods have a reduced first order temporal accuracy due to the semi-explicit treatment of boundary constraints.

2. Background: strongly-coupled immersed boundary formulation

In this section, we first review the strongly-coupled immersed boundary (IB) formulation by Goza and Colonius [15], discuss the source of the computational bottleneck encountered by this approach, and demonstrate the appearance of this bottleneck in other semi-strongly coupled and strongly coupled numerical methods. In the next section, we will discuss the remedy to this bottleneck.

2.1. Governing equations

We consider a fluid domain Ω and a set of immersed bodies Γ. We present a formulation for FSI problems involving a collection of m_r rigid bodies, Γ_r^i for $i = 1, \ldots, m_r$, along with m_t torsional bodies Γ_t^i for $i = 1, \ldots, m_t$ and m_d deformable bodies Γ_d^i for $i = 1, \ldots, m_d$. The torsional bodies are assumed to be mounted on some subset of the rigid bodies, as shown in Fig. 1. The readers are referred to [14] for details about bodies that are torsionally connected to other torsional bodies. Incorporating this extension would involve only superficial changes to the formulation. The dimensionless governing equations are written as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \frac{1}{Re} \nabla^2 \boldsymbol{u} + \int_{\Gamma} \boldsymbol{f}(\boldsymbol{\chi}(s,t)) \delta(\boldsymbol{\chi}(s,t) - \boldsymbol{x}) ds \tag{1}$$

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

$$i_t^i \frac{\partial^2 \theta^i}{\partial t^2} + c_t^i \frac{\partial \theta^i}{\partial t} + k_t^i \theta^i = -\int_{\Gamma_t^i} (\boldsymbol{\chi}_t^i - \boldsymbol{\chi}_t^{0i}) \times \boldsymbol{f}(\boldsymbol{\chi}_t^i) d\boldsymbol{\chi}_t^i + g_t(\theta^i) \qquad \text{for} \quad i = 1, \ldots, m_t \tag{3}$$

$$\frac{\rho_d^i}{\rho_f} \frac{\partial^2 \boldsymbol{\chi}_d^i}{\partial t^2} = \frac{1}{\rho_f U_\infty^2} \nabla \cdot \boldsymbol{\sigma}^i + \boldsymbol{g}_d(\boldsymbol{\chi}_d^i) - \boldsymbol{f}(\boldsymbol{\chi}_d^i) \qquad \text{for} \quad i = 1, \ldots, m_d \tag{4}$$

$$\int_{\Omega} \boldsymbol{u}(\boldsymbol{x}) \delta(\boldsymbol{x} - \boldsymbol{\chi}_r^i) d\boldsymbol{x} = \boldsymbol{u}_r^i(\boldsymbol{\chi}_r^i) \qquad \text{for} \quad i = 1, \ldots, m_r \tag{5}$$

$$\int_{\Omega} \boldsymbol{u}(\boldsymbol{x}) \delta(\boldsymbol{x} - \boldsymbol{\chi}_t^i) d\boldsymbol{x} = \frac{\partial \theta^i}{\partial t} \hat{\boldsymbol{e}}^i \times (\boldsymbol{\chi}_t^i - \boldsymbol{\chi}_t^{0i}) \qquad \text{for} \quad i = 1, \ldots, m_t \tag{6}$$

$$\int_{\Omega} \boldsymbol{u}(\boldsymbol{x}) \delta(\boldsymbol{x} - \boldsymbol{\chi}_d^i) d\boldsymbol{x} = \frac{\partial \boldsymbol{\chi}_d^i}{\partial t} \qquad \text{for} \quad i = 1, \ldots, m_d \tag{7}$$

In the above, $\boldsymbol{x}$ denotes the Eulerian coordinate representing a position in space and $\boldsymbol{\chi}(s,t)$ denotes the Lagrangian coordinate attached to the bodies in the set Γ, the surface of which is parametrized by the variable s. These variables, $\boldsymbol{x}$, $\boldsymbol{\chi}$ and s were nondimensionalized by a characteristic length scale L; velocity $\boldsymbol{u}$ was nondimensionalized by a characteristic velocity scale U_∞; time t was nondimensionalized by L/U_∞; pressure p and surface stress imposed on the fluid by the body $\boldsymbol{f}$ were nondimensionalized by $\rho_f U_\infty^2$, where ρ_f is the fluid density. The Reynolds number in Eq. (1) is defined as $Re = U_\infty L/\nu$, where ν is the kinematic viscosity of the fluid.

The equation of motion of the i^{th} torsional body Γ_t^i is given by Eq. (3) where θ^i is the deflection angle of the body from its undeformed configuration θ^{0i} and $\boldsymbol{\chi}_t^i$ is the Lagrangian coordinate of Γ_t^i. Here, i_t^i denotes the moment of inertia of the torsionally connected body about a hinge location $\boldsymbol{\chi}_t^{0i}$ nondimensionalized as $i_t^i = I_t^i/\rho_f L^4$, where I_t^i is the dimensional moment of inertia. Similarly, the torsional spring has a nondimensional stiffness $k_t^i = K_t^i/\rho_f U_\infty^2 L^2$ and damping coefficient $c_t^i = C_t^i/\rho_f U_\infty L^3$, where K_t^i and C_t^i are the dimensional quantities, respectively. The first term on the right hand side of Eq. (3) represents the moment about $\boldsymbol{\chi}_t^{0i}$ due to the surface stress imposed on the fluid by the body (thereby resulting in a negative sign). The second term g_t represents moments due to body forces such as gravity, pseudo forces, *etc.*

The equation of motion of the i^{th} deformable body Γ_d^i is given by Eq. (4) where $\boldsymbol{\chi}_d^i$ is the Lagrangian coordinate of Γ_d^i. Here ρ_d^i is the density of the structure, $\boldsymbol{\sigma}^i$ is the Cauchy stress tensor contributing to the internal restoring forces of the body and $\boldsymbol{g}_d$ denotes the body force per unit volume due to gravity, pseudo forces etc. See reference [15] for a detailed description about these quantities.

The no-slip boundary constraints on the rigid, torsional and deformable bodies are given by Eq. (5), (6) and (7), respectively. Here, $\boldsymbol{u}_r^i$ is the (possibly zero) prescribed velocity on the rigid body Γ_r^i, and $\hat{\boldsymbol{e}}^i$ is a unit vector denoting the direction of the angular velocity of the torsional body Γ_t^i. These no-slip constraints are used to solve for the surface stress term $\boldsymbol{f}(\boldsymbol{\chi})$ that enforces the boundary condition that must hold on the respective bodies.

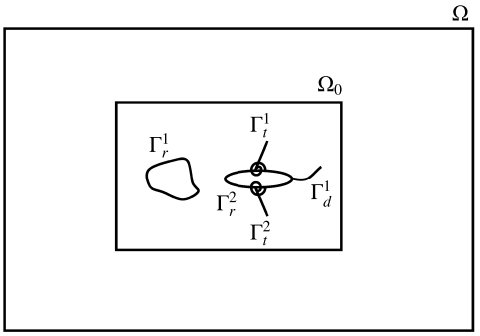


Fig. 1. Schematic of the computational domain consisting of the flow domain, Ω and five immersed bodies, $\Gamma = \{\Gamma_r^1, \Gamma_r^2, \Gamma_t^1, \Gamma_t^2, \Gamma_d^1\}$. Rigid bodies include Γ_r^1 and Γ_r^2 while Γ_t^1 and Γ_t^2 are torsional bodies and Γ_d^1 is a deformable body mounted on Γ_r^2. The sub-domain of our proposed approach described in Sec. 3 that encompasses the range of motion of all bodies is denoted by Ω_0.

2.2. Fully discretized equations

Following [15], Eq. (1) is spatially discretized using the standard second-order finite difference operators on a staggered grid and rewritten in a streamfunction-vorticity formulation. A finite element procedure described in [15] is used to spatially discretize Eq. (4). For time-discretization, the flow equations (1) utilize an Adams-Bashforth scheme for the nonlinear term and a Crank-Nicolson method for the diffusive term. The structural equations of motion (3) and (4) are discretized using an implicit Newmark scheme. The boundary conditions (5)–(7) and the surface stress term in Eq. (1) are evaluated implicitly at the current time step to enable stability of the method for bodies with a wide range of mass ratio and undergoing large body displacements. The fully discretized equations are given below,

$$C^T A C s_{n+1} + C^T E_{n+1}^T f_{n+1} = r_n^f \tag{8}$$

$$\frac{4}{\Delta t^2} i_t^i \theta_{n+1}^i + \frac{2}{\Delta t} c_t^i \theta_{n+1}^i + k_t^i \theta_{n+1}^i - Q_t^i R_{t,n+1}^i f_{t,n+1}^i \Delta s = r_n^{\phi,i} \quad \text{for} \quad i = 1, \ldots, m_t \tag{9}$$

$$\frac{2}{\Delta t} \theta_{n+1}^i - \phi_{n+1}^i = r_n^{\theta,i} \quad \text{for} \quad i = 1, \ldots, m_t \tag{10}$$

$$\frac{4}{\Delta t^2} M_d^i \chi_{d,n+1}^i + R_d^i(\chi_{d,n+1}^i) - Q_d^i W_{d,n+1}^i f_{d,n+1}^i = r_n^{\zeta,i} \quad \text{for} \quad i = 1, \ldots, m_d \tag{11}$$

$$\frac{2}{\Delta t} \chi_{d,n+1}^i - \zeta_{d,n+1}^i = r_n^{\chi,i} \quad \text{for} \quad i = 1, \ldots, m_d \tag{12}$$

$$E_{r,n+1}^i C s_{n+1} = u_{r,n+1}^i \quad \text{for} \quad i = 1, \ldots, m_r \tag{13}$$

$$E_{t,n+1}^i C s_{n+1} - R_{t,n+1}^{iT} Q_t^{iT} \phi_{n+1}^i = 0 \quad \text{for} \quad i = 1, \ldots, m_t \tag{14}$$

$$E_{d,n+1}^i C s_{n+1} - \zeta_{d,n+1}^i = 0 \quad \text{for} \quad i = 1, \ldots, m_d \tag{15}$$

Here, the subscript n denotes the time step; the discrete streamfunction and surface stresses imposed on all bodies by fluid are denoted by s and f, respectively; the stresses on the individual torsional and deformable bodies are denoted by f_t^i, $f_d^i \in f$, respectively (there is also a set of surface stresses associated with the rigid bodies, $f_r^i \in f$). We also define $\phi^i = \dot{\theta}^i$ and $\zeta_d^i = \dot{\chi}_d^i$. The curl operator is given by C; $A = \frac{1}{\Delta t} I - \frac{1}{2} L$ where Δt is the time step size, I is the identity and L is the vector Laplacian operator. The discretization of the operators in the left hand side of Eq. (5)–(7), E_r^i, E_t^i, and E_d^i, are the IB interpolation operators that interpolate the fluid velocity onto the rigid, torsional and deformable bodies, respectively and E is simply the block-row aggregation of each of E_r^i, E_t^i, and E_d^i. On the other hand, E^T represents the regularization operator involving the delta function in Eq. (1) which regularizes surface stress from each of the bodies onto the flow field. See reference [3] for more details about the standard finite volume discretizations used to represent fluid operators (e.g., C, L) as well as more information on the IB interpolation and regularization operators.

The operator $Q_t^i R_{t,n+1}^i$ denotes the discretization of the term involving the surface stress in Eq. (3) and Δs is the size of discretization of the body while M_d^i, R_d^i and $Q_d^i W_{d,n+1}^i$ are the finite element operators corresponding to the first, second and fourth terms of Eq. (4). See Appendix A for the more details of these operators. The expressions of the right hand side terms r_n^f, $r_n^{\phi,i}$, $r_n^{\theta,i}$, $r_n^{\zeta,i}$ and $r_n^{\chi,i}$, which are the known right-hand side quantities that arise from the explicit temporal treatment and boundary conditions, are also provided in Appendix A.

2.3. Algorithm for strong fluid-structure coupling and associated computational bottleneck

The implicit treatment of body variables and the no-slip constraint in strongly-coupled IB methods necessitates an iterative method to solve the above system of equations (8)–(15). However, a straightforward implementation of iterating all the equations until convergence will incur significant expense since the flow equations scale by the large number of flow points. An observation of Goza and Colonius [15] was that Eq. (8)–(15) can be subjected to a block-LU decomposition before applying an iterative scheme so that the iterations are only restricted to the evaluation of small-dimensional systems that scale only with the number of points on the immersed surface. The full derivation of this procedure is provided in Appendix A for self-containment. The final system of LU-factored equations is

$$s^* = (C^T A C)^{-1} r_n^f \tag{16}$$

$$\left(E_{n+1}^{(k)} C (C^T A C)^{-1} C^T E_{n+1}^{(k)T} + \frac{2\Delta s}{\Delta t} S_{t,n+1}^{(k)T} J_t^{-1} S_{t,n+1}^{(k)} + \frac{2}{\Delta t} \hat{I}_d^T {J_d^{(k)}}^{-1} S_{d,n+1}^{(k)}\right) f_{n+1}^{(k+1)} = E_{n+1}^{(k)} C s^* - r^{c(k)} + S_{t,n+1}^{(k)T}\left(r^{\theta(k)} - \frac{2}{\Delta t} J_t^{-1} r^{\phi(k)}\right) + \hat{I}_d^T\left(r^{\chi(k)} - \frac{2}{\Delta t} {J_d^{(k)}}^{-1} r^{\zeta(k)}\right) \tag{17}$$

$$\Delta\theta = J_t^{-1}\left(r^{\phi(k)} + \Delta s S_{t,n+1}^{(k)} f_{n+1}^{(k+1)}\right) \tag{18}$$

$$\Delta\chi_d = {J_d^{(k)}}^{-1}\left(r^{\zeta(k)} + S_{d,n+1}^{(k)} f_{n+1}^{(k+1)}\right) \tag{19}$$

$$s_{n+1} = s^* - (C^T A C)^{-1} C^T E_{n+1}^T f_{n+1} \tag{20}$$

Here, the superscript (k) denotes the FSI iteration. $S_{t,n+1}^{(k)}$, $S_{d,n+1}^{(k)}$, J_t, $J_d^{(k)}$ and $\hat{I}_d$ are the aggregated block-diagonal matrices containing the individual structural operators with the superscript i in (8)-(15). The expressions of these block-diagonal operators as well as the right-hand side terms r_n^f, $r^{c(k)}$, $r^{\theta(k)}$, $r^{\phi(k)}$, $r^{\chi(k)}$ are provided in Appendix A.

Now, the entire method can be efficiently divided into three steps. First, a trial streamfunction s^* is predicted without accounting for the body forces in Eq. (16). Next, the FSI coupling Eq. (17)–(19) are solved iteratively at the next time step $n+1$ for the surface stress $f_{n+1}^{(k+1)}$ and body configuration $\theta_{n+1}^{(k+1)} = \theta_{n+1}^{(k)} + \Delta\theta$, $\chi_{d,n+1}^{(k+1)} = \chi_{d,n+1}^{(k)} + \Delta\chi_d$. Within each FSI iteration, the linear system in Eq. (17) is solved using an iterative method such as GMRES. We note that there is a distinction between the FSI iterations associated with Eq. (17)–(19) and the GMRES iterations used to solve Eq. (17) within each FSI iteration. We will differentiate between these two types of iterations as needed for clarity of context. Finally, the streamfunction at the current time step, s_{n+1}, is obtained by correcting s^* using the updated surface stress in Eq. (20).

We note that the trial and corrected streamfunctions and therefore Eq. (16) and (20) scale by the large number of flow points. However, Eq. (16) and (20) do not depend on the FSI iterate, k, and therefore, are solved only once at the beginning and end of the time-step, respectively. These steps thus incur the same cost as compared to the non-FSI, stationary body case, which is a lower bound for the computational expense one can expect to obtain for fully coupled FSI simulations. In contrast, Eq. (17)–(19) are solved for multiple FSI iterates k within a single time step. Since the system of iterated equations (17)–(19) is small dimensional scaling by the number body points, nominally a significant amount of computational savings can be expected as compared to a straightforward implementation of iterating over all the equations.

These savings are realized due to the block-LU decomposition of Eq. (8)–(15). However, an undesirable consequence of this decomposition procedure is that a large linear system in the form on $(C^T A C)^{-1}$ that scales with the number of points in the flow domain, n_s, gets embedded within the small dimensional matrix in Eq. (17), $E_{n+1}^{(k)} C (C^T A C)^{-1} C^T E_{n+1}^{(k)T}$. Merely constructing the small-dimensional matrix $E_{n+1}^{(k)} C (C^T A C)^{-1} C^T E_{n+1}^{(k)T}$ is computationally expensive since it requires several large computations involving $(C^T A C)^{-1}$. Furthermore, this small matrix depends on the time-varying position of the immersed body, and therefore changes at least once per FSI iteration within each time step.

The full construction of $E_{n+1}^{(k)} C (C^T A C)^{-1} C^T E_{n+1}^{(k)T}$ may be circumvented by a matrix-free implementation of GMRES. However, even in the matrix-free implementation, these $(C^T A C)^{-1}$ operations are performed once in every GMRES iteration within every FSI iteration. For example, if the algorithm requires 3 FSI iterations per time step and on average, each FSI iteration requires 5 GMRES iterations, then a total of $3 \times 5 = 15$ operations of $(C^T A C)^{-1}$ are performed just in a single time step. Therefore, even though the underlying linear system in (17) is small dimensional, multiple solves of the large embedded system is inevitable.

The root cause for this bottleneck is the need to solve the system of equations (8)–(15) simultaneously arising from the implicit treatment of body forces, positions and no-slip constraint in strongly-coupled methods. This implicit treatment necessitates the computation of the surface stress such that it enforces the no-slip constraint at the current time step. In the IB method of Goza and Colonius [15], this implicit treatment is manifested in the first term of Eq. (17) as

$$\overbrace{E_{n+1}^{(k)}C}^{\text{Contribution to no-slip velocity}} \quad \overbrace{(C^TAC)^{-1}}^{\text{Globally affected flow-field}} \quad \overbrace{C^TE_{n+1}^{(k)T}f_{n+1}^{(k+1)}}^{\text{Local fluid source}} \tag{21}$$

A source term in the form of surface stress $f_{n+1}^{(k+1)}$ is converted into a local fluid source in the vicinity of the Lagrangian body points via the action of $C^TE_{n+1}^{(k)T}$. Then the elliptic Poisson-like operator $(C^TAC)^{-1}$ containing the viscous contribution globally modifies the flow-field. Finally, the no-slip velocity on the body enforced by the surface stress is obtained via interpolation of the globally affected flow-field via $E_{n+1}^{(k)}C$.

We emphasize that the above-mentioned bottleneck of solving a large-dimensional system for the small dimensional body variables is not limited to the IB method of Goza and Colonius [15]. Broadly speaking, fully implicit, strongly coupled IB methods require iterations to arrive at a solution that satisfies the flow and structural equations of motion as well as the nonlinear no-slip constraint. Many iterative approaches require iterating on all flow (velocity, pressure) and structural (displacement, forces) variables, the former of which requires the solution of large-dimensional systems that scale with the number of flow points [11,13,12]. Other approaches more similar to that of Goza and Colonius [15] are able to reformulate the discrete equations, through either a block Gauss-Seidel approach [14] or a block-LU factorization [16], so that any required iterations are restricted to nominally small dimensional systems in analogy with (17)–(19). However, similar to the algorithm of Goza and Colonius [15], these small dimensional systems that scale with the number of body points at the fluid-structure interface have embedded large linear systems that scale with the number of points in the flow domain.

In the next section, we propose an efficient algorithm that addresses the above-mentioned bottleneck of all strongly coupled and some semi-strongly coupled IB methods. Our proposed approach leverages the block-LU factored form of the equations (16)–(20), so that the FSI iterations are restricted to small dimensional systems. However, it can be also extended to other strongly coupled algorithms which are able to reformulate the equations to restrict the FSI iterations to such small dimensional systems. We provide a strategy to precompute the matrix that encapsulates the large linear system on a sub-domain that envelops the full range of structural motion. The matrix is then updated to accurately enforce the no-slip constraint via interpolation onto the portion of the sub-domain for the location of the current structures. The precomputation procedure avoids additional large linear solves (compared to the non-FSI, stationary body case) while marching in time, while the interpolation procedure allows for accurate treatment of arbitrarily large structural motions.

3. Proposed approach for treating arbitrarily moving bodies as efficiently as stationary bodies

Our proposed idea is motivated from the observation that for *stationary* bodies, the operator $E_{n+1}^{(k)}$ and therefore, the small-dimensional operator $E_{n+1}^{(k)}C(C^TAC)^{-1}C^TE_{n+1}^{(k)T}$ do not vary in time. This allows one to compute $E_{n+1}^{(k)}C(C^TAC)^{-1}C^TE_{n+1}^{(k)T}$ once and for all, thereby circumventing the need to compute the computationally expensive $(C^TAC)^{-1}$ at every GMRES iteration within each time step. Similarly, to avoid computing $(C^TAC)^{-1}$ multiple times in Eq. (17) for *non-stationary* bodies, we propose the following approximation for $E_{n+1}^{(k)}$,

$$E_{n+1}^{(k)} \approx P_{n+1}^{(k)}E_0 \tag{22}$$

where E_0 is an IB interpolation operator similar to $E_{n+1}^{(k)}$, but defined on a *sub-domain* defined as a fixed set of n_{sd} Eulerian points in the flow domain, $\Omega_0 \subset \Omega$ as shown in Fig. 1. These sub-domain points are selected *a priori*, independent of the time-instantaneous body locations, and therefore E_0 is time-invariant. The actual IB interpolation operator $E_{n+1}^{(k)}$ defined on the moving Lagrangian body points is then recovered by the application of an interpolation operator $P_{n+1}^{(k)}$ (not the same as the IB interpolation operator E) on E_0. This operator $P_{n+1}^{(k)}$ is time varying but may be evaluated sparsely and cheaply, as it only involves a small number of nonzero interpolation weights near the various structural interfaces (contained within the sub-domain). More details about E_0 and $P_{n+1}^{(k)}$ are discussed in Sec. 3.1. For now, the previously expensive operation of Eq. (17) can be rewritten as,

$$E_{n+1}^{(k)}C(C^TAC)^{-1}C^TE_{n+1}^{(k)T} \approx P_{n+1}^{(k)}\left(E_0C(C^TAC)^{-1}C^TE_0^T\right)P_{n+1}^{(k)T} = P_{n+1}^{(k)}BP_{n+1}^{(k)T} \tag{23}$$

where $B = E_0C(C^TAC)^{-1}C^TE_0^T$. The above reformulation facilitates the following:

a. Since B is time-invariant and scales by a size smaller than the flow points, $n_{sd} < n_s$ (often $n_{sd} \ll n_s$ depending on the range of the bodies' motions), we can compute and store B once and for all, thereby circumventing multiple $(C^TAC)^{-1}$ operations.
b. Additionally, since $P_{n+1}^{(k)}$ is sparse, evaluation of $P_{n+1}^{(k)}BP_{n+1}^{(k)T}$ is performed at minimal computational cost that scales only with a small multiple of the number of body interface points.

3.1. Sub-domain IB interpolation operator, E_0, and sparse interpolation operator P

The IB interpolation operator is constructed from the regularized discrete delta-function [1,21]. If we denote the discrete delta function as $d(\cdot)$, then the interpolation operator for interpolating the velocity from a Eulerian flow point at $\boldsymbol{x} = (x_j, y_j)$ to a Lagrangian body point $\boldsymbol{\chi} = (\xi_i, \eta_i)$ is given (to within a scaling factor [3]) by,

$$E_{ij} \equiv d(x_j - \xi_i)\, d(y_j - \eta_i) \tag{24}$$

Note that the time subscript $n+1$ and iteration superscript k are dropped for neatness. In our proposed approach, however, we first define a sub-domain $\Omega_0 \subset \Omega$ as shown in Fig. 1 and define an associated set of points $\boldsymbol{x}^0 \in \Omega_0$. The procedure for selecting the sub-domain is provided in Sec. 3.1.1. In contrast to Eq. (24), the sub-domain interpolation operator is now defined between the Eulerian flow point $\boldsymbol{x} = (x_j, y_j)$ and Eulerian flow sub-domain point $\boldsymbol{x}^0 = (x_i^0, y_i^0)$ as,

$$E_{0ij} \equiv d(x_j - x_i^0)\, d(y_j - y_i^0) \tag{25}$$

This interpolation operator associated with the sub-domain Ω_0, (25), may be precomputed at the fixed set of sub-domain points. The desired IB interpolation operator, E, associated with the time varying body locations is then approximated through interpolation of the precomputed sub-domain interpolation operator, E_0 via

$$E_{ij} \equiv d(x_j - \xi_i)\, d(y_j - \eta_i) \approx \sum_{k=1}^{n_{sd}} w_{ik}\, d(x_j - x_k^0)\, d(y_j - y_k^0) \equiv \sum_{k=1}^{n_{sd}} P_{ik}\, E_{0kj} \tag{26}$$

where w_{ik} are the weights of interpolation which are stored in the operator P.

The expression (26) is meant to be illustrative of the interpolation process. In practice, it is wasteful to utilize the entire sub-domain Ω_0 to construct the interpolation weights. Instead, we perform local interpolation using only a small number of $n_p \ll n_{sd}$ nearest neighboring sub-domain points to the body-point. In particular, for approximating E_{ij} at the i^{th} body point (ξ_i, η_i), we identify n_p nearest neighboring sub-domain points $(x_{i_k}^0, y_{i_k}^0)$ where $i_k \in \{1, \ldots, n_{sd}\}$ for $k = 1, \ldots, n_p$. In other words, $(x_{i_k}^0, y_{i_k}^0)$ represents the k^{th} nearest neighbor point on the sub-domain associated with the i^{th} body point (ξ_i, η_i). Accordingly, the interpolation in Eq. (26) can be locally performed as

$$E_{ij} \equiv d(x_j - \xi_i)\, d(y_j - \eta_i) \approx \sum_{k=1}^{n_p} w_{ii_k}\, d(x_j - x_{i_k}^0)\, d(y_j - y_{i_k}^0) \equiv \sum_{k=1}^{n_p} P_{ii_k}\, E_{0i_kj} \tag{27}$$

where now only w_{ii_k} needs to be stored for the body index i and $w_{il} = 0 \;\; \forall\, l \in \{1, \ldots, n_{sd}\}, l \neq i_k$ for $k = 1, \ldots, n_p$.

In this way, we may consider only the number of nearest neighbors, n_p, in constructing and applying P, rather than the total number of points in the sub-domain, n_{sd}. This formulation allows for P, which is time dependent, to be efficiently constructed and applied via sparse operations. The procedure for identifying the n_p sub-domain points nearest to a body point is provided in Sec. 3.1.3. We also show in Appendix B that the interpolated delta functions in the right hand side of the above equation satisfy zeroth to third (and potentially more) moment conditions with at least second order accuracy provided that the underlying delta functions being interpolated also satisfy these conditions. This minimum second order accuracy is in accordance with the maximum second order accuracy of the immersed boundary method.

Finally, we emphasize that the operator B is constructed for all the points in the sub-domain since it embeds the sub-domain interpolation operator E_0. Therefore, when the body point moves to a different location or finite volume cell, only the nearest neighboring sub-domain points change and the operator P is reconstructed, albeit sparsely. However, the previously constructed sub-domain, associated IB interpolation operator E_0 and the precomputed operator B remain unchanged.

3.1.1. Procedure for selecting a sub-domain

First, a rectangular sub-domain as shown in Fig. 1 is considered for simplicity. Next, the sub-domain boundaries are chosen such that all the bodies are guaranteed *a priori* to stay within the sub-domain at all time instants. This can be achieved by examining the physical displacement limits of the body and total simulation time. A physical intuition of the problem can also help in choosing a more compact sub-domain. Since choosing the sub-domain is problem dependent, it will be discussed in more detail for specific problems in Sec. 5. Next, the grid spacing between the sub-domain points is set to be equal to the flow grid spacing. This choice was observed to provide accurate results for the set of problems considered in Sec. 5. Furthermore, in the staggered grid configuration, the sub-domain points are chosen to coincide with the vorticity points on cell vertices, as shown in Fig. 5, so that the sub-domain points are equidistant from the $x-$ and $y-$ velocity points located on the cell edges.

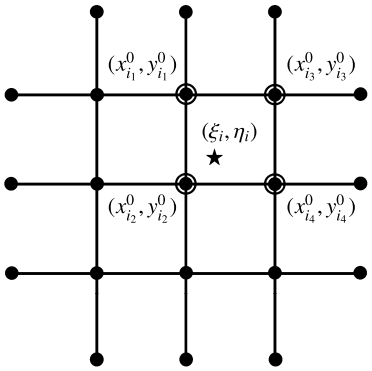


Fig. 2. Schematic for identifying $n_p = 4$ nearest neighboring sub-domain points $(x^0_{i_k}, y^0_{i_k})$ for the body point (ξ_i, η_i).

3.1.2. Choice of interpolation method

A variety of interpolation functions such as Lagrange interpolation functions, delta functions, polynomial functions *etc.*, can be used for performing interpolation and constructing P in Eq. (27). Since the use of delta functions for constructing E is well known and studied in the IB framework, we use delta functions for constructing P as well. We will denote these delta functions as $d_p(\cdot)$ and emphasize that the discrete delta functions $d_p(\cdot)$ used in P may be different from $d(\cdot)$ used for constructing E. While the choice of $d(\cdot)$ is governed by the need to regularize and remove unphysical oscillations in surface stress [22], $d_p(\cdot)$ is chosen to strike a balance between the sparsity of P and accuracy of interpolation.

In this work, we use a two-point hat function [23] given by

$$d_p(r) = \begin{cases} 1 - \frac{|r|}{\Delta r}, & |r| < \Delta r \\ 0, & |r| > \Delta r \end{cases} \tag{28}$$

where Δr is the flow sub-domain grid spacing in the r-direction. We choose this delta function because it has a support of only one cell and yet it is $\mathcal{O}((\Delta x^0)^2)$ accurate where Δx^0 is the sub-domain grid spacing. A single cell support implies that for two dimensional IB method, only $n_p = 4$ input points are required for interpolation, thereby, enabling an extremely sparse construction of P with only $n_p = 4$ non-zeros per row. Furthermore, we note that the second order interpolation method does not affect the original first order spatial accuracy [20] of projection based immersed boundary methods. Now, the weights of interpolation in Eq. (27) are given by,

$$w_{ii_k} = d_p(\xi_i - x^0_{i_k})\, d_p(\eta_i - y^0_{i_k}) \tag{29}$$

3.1.3. Identification of sub-domain points for local interpolation

For local interpolation, $n_p = 4$ nearest neighboring sub-domain points that form a tensor grid are identified. For instance, consider the sub-domain points in a two-dimensional space denoted by '•' as shown in Fig. 2. For approximating the operator E at the body point (ξ_i, η_i) denoted by '⋆', the four nearest neighboring points $(x^0_{i_k}, y^0_{i_k})$ that form a tensor grid denoted by '∘' are identified.

3.2. Approximating B as a sparse operator

The proposed sub-domain approach requires precomputing and storing the operator B. However, we note that B is a dense matrix and therefore, storing B can become computationally prohibitive for problems with large sub-domain and fine grid discretization. To circumvent this computational storage issue, we note that B has a compact structure and therefore, we approximate the dense B operator as sparse. In Sec. 3.2.1, we provide justification that B can be indeed constructed sparsely up to a drop tolerance. Then a drop tolerance filtering technique similar to that employed in incomplete LU decomposition [24] to construct B sparsely is provided in Sec. 3.2.2.

3.2.1. Analysis of compactness of B

For clarity, we specify the dimensions of the previously defined operators as $E_0 \in \mathbb{R}^{2n_{sd} \times n_q}$, $B \in \mathbb{R}^{2n_{sd} \times 2n_{sd}}$, $C \in \mathbb{R}^{n_q \times n_s}$ and $A \in \mathbb{R}^{n_q \times n_q}$, respectively, where n_{sd}, n_q and n_s are the number of sub-domain grid points, sum of velocity grid points in x and y coordinate directions $(n_u + n_v)$, and vorticity grid points, respectively.

Firstly, we will focus on the interior term $C(C^TAC)^{-1}C^T$ of $B = E_0C(C^TAC)^{-1}C^TE_0^T$. Since $A = I_q + \alpha CC^T$, where $\alpha = \frac{\Delta t}{2Re\Delta x^2}$, $I_q \in \mathbb{R}^{n_q \times n_q}$ is the identity and $CC^T \in \mathbb{R}^{n_q \times n_q}$ is the 2D vector Laplacian matrix, C^TAC can be rewritten as,

$$C^TAC = C^TC(I_s + \alpha C^TC) \tag{30}$$

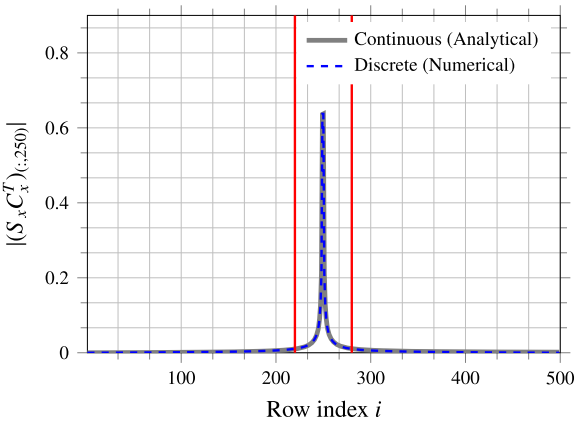


Fig. 3. Plot of the 250^{th} column of the discrete $|S_x C_x^T|$ obtained numerically via Fourier transforms and the continuous $|S_x C_x^T|$ obtained analytically from Eq. (33). The region between the red lines indicates the non-zero locations retained when applying a relative drop tolerance of 10^{-2} relative to $L/2$ for the continuous part and 1 for the discrete part. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

where $I_s \in \mathbb{R}^{n_s \times n_s}$ is the identity and $C^T C \in \mathbb{R}^{n_s \times n_s}$ is the standard 2D scalar Laplacian. $C^T C$ can be diagonalized as $C^T C = S \Lambda S^T$, where the eigenvectors $S \in \mathbb{R}^{n_s \times n_s}$ are the discrete sine transforms and $\Lambda \in \mathbb{R}^{n_s \times n_s}$ contains the eigenvalues. Accordingly, we can define the singular value decomposition, $C = U_c \Lambda^{1/2} S^T$ where $U_c \in \mathbb{R}^{n_q \times n_s}$ is the left singular vector. On substituting these decompositions, we get for the interior term,

$$C(C^T A C)^{-1} C^T = U_c (I_s + \alpha \Lambda)^{-1} U_c^T \tag{31}$$

For a conservative choice of grid Reynolds number $Re \Delta x = 1$ and time discretization $\Delta t = \Delta x/4$ resulting in $\alpha = 0.125$, $I_s + \alpha \Lambda$ has a small condition number of 2. Therefore, we note that $I_s + \alpha \Lambda$ is nearly a constant diagonal matrix (less conservative grid Reynolds numbers would only act to improve this approximation). Thus, $U_c U_c^T$ will have nearly the same sparsity structure as that of $U_c (I_s + \alpha \Lambda)^{-1} U_c^T$. We therefore demonstrate below that $U_c U_c^T$ is well approximated as a sparse matrix, and use this to argue that the latter matrix $U_c (I_s + \alpha \Lambda)^{-1} U_c^T$ will also be sparse, to within mild changes in sparsity pattern and index due to the slight non-unity condition number. We therefore show in this section that the matrix of interest can be expected to be sparse, and subsequently introduce a drop tolerance technique in Sec. 3.2.2 to identify which nonzero entries to retain.

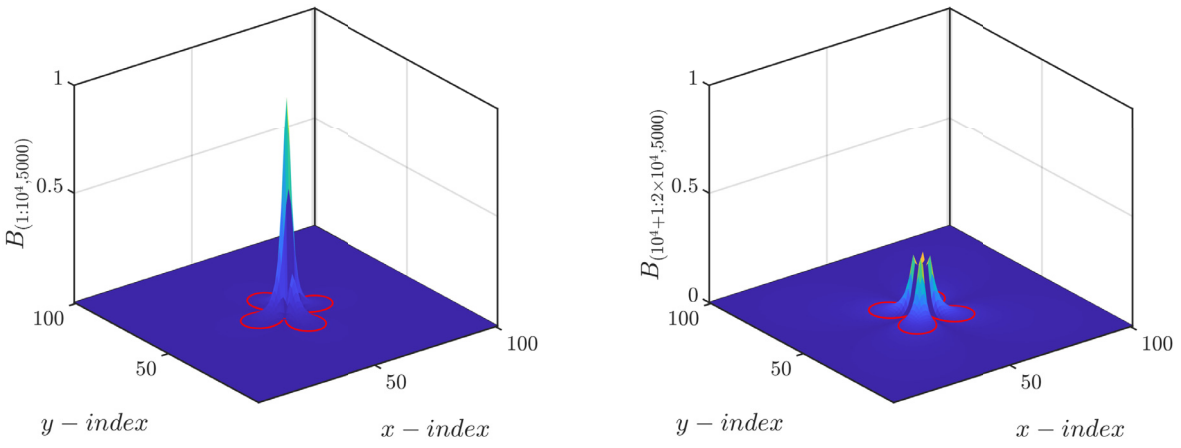
We note that U_c is comprised of eigenvectors of the 2D vector Laplacian, $C C^T = U_c \Lambda U_c^T$, that mimics $\nabla^2 \boldsymbol{u}$. In Cartesian coordinates, $\nabla^2 \boldsymbol{u}$ reduces to the scalar Laplacian applied to each velocity component. Therefore, we can segregate U_c as $U_c \equiv [F_u, F_v]^T$ where $F_u \in \mathbb{R}^{n_u \times n_s}$ and $F_v \in \mathbb{R}^{n_v \times n_s}$ are the eigenvectors of the scalar Laplacian acting on the x and y velocities, u and v, respectively. On staggered grids, u and v have mixed boundary conditions on cell faces to enforce zero vorticity conditions on cell vertices. For instance, homogeneous Dirichlet boundary conditions in the x-direction and Neumann boundary conditions in the y-direction are imposed on the u-velocity and vice versa for v-velocity. Therefore, U_c contains a mixture of sines and cosines – $F_u = S_x \otimes C_y$ and $F_v = C_x \otimes S_y$, where $\otimes$ denotes the Kronecker product, S_x and S_y are 1D discrete sine transforms (type-I) and C_x and C_y are 1D discrete cosine transforms (type-II, excluding the constant $[1, \ldots, 1]^T$ vector that spans the null space of the Neumann operator). On substituting these decompositions we obtain

$$U_c U_c^T \equiv \begin{bmatrix} S_x S_x^T \otimes C_y C_y^T & S_x C_x^T \otimes C_y S_y^T \\ C_x S_x^T \otimes S_y C_y^T & C_x C_x^T \otimes S_y S_y^T \end{bmatrix} \tag{32}$$

Here, the block diagonal entries are approximately identity because the sines and cosines are mutually orthogonal among themselves. Note that they are not *exactly* identity because the discrete cosine vectors are truncated by one due to the exclusion of the constant null space vector. On the other hand, for the off-diagonal block terms, consider for instance, the continuous counterpart of the $(i, j+1)$ component of $S_x C_x^T$,

$$(S_x C_x^T)_{(i,j+1)} = (S_x^T C_x^{(3)})_{(i,j+1)} \xrightarrow[\text{analog}]{\text{continuous}} \int_0^L \sin \frac{i\pi x}{L} \cos\left(j + \frac{1}{2}\right) \frac{\pi x}{L} dx = \frac{L}{\pi(2i - 2j - 1)} + \frac{L}{\pi(2i + 2j + 1)} \tag{33}$$

where $C_x^{(3)} = C_x^T$ is the discrete cosine transform of type-III [25]. From Eq. (33) it can be seen that any row or column of $|S_x C_x^T|$ is the discrete analog to a quantity that decays as $\frac{1}{|i-j|}$ with a peak when $i = j$. For reference, consider a problem with grid dimensions 500×500, for which we plot in Fig. 3 the 250^{th} column of the discrete $|S_x C_x^T|$ obtained numerically via Fourier transforms and the continuous $|S_x C_x^T|$ obtained analytically from Eq. (33). Note that for plotting the analytical



(a) Contour plot of the first 10^4 rows.

(b) Contour plot of the last 10^4 rows.

Fig. 4. Plot of the 5000^{th} column of B constructed for a sub-domain of grid dimensions 100×100 normalized with respect to the maximum absolute value of B in that column. The regions within the red contour lines indicate the non-zero locations retained when applying a drop tolerance of 10^{-2} relative to that maximum absolute value.

part, the right most expression from Eq. (33) is scaled (multiplied) by $1/(L/2)$, since $L/2$ is the value corresponding to the continuous counterpart of the diagonal of $S_x S_x^T$. The discrete $|(S_x C_x^T)_{(:,250)}|$ decays similarly to its continuous counterpart as $\frac{1}{|i-j|}$. The decay rate is increased as $\frac{1}{|i-j||k-l|}$ when we consider the entire off-diagonal block $S_x C_x^T \otimes C_y S_y^T$, where k and l indices correspond to the $(k+1, l)$ component of $C_y S_y^T$. Similar decaying trends can be derived for the remaining off-diagonal block $C_x S_x^T \otimes S_y C_y^T$. Under a drop tolerance filtering criteria where the matrix elements below a specified relative tolerance be dropped to zero, these off-diagonal blocks can be approximated sparsely.

To indicate the impact of applying this drop-tolerance filtering procedure, a tolerance of 10^{-2} relative to $L/2$ will result in retaining approximately 60 non-zeros per row for $S_x C_x^T$ irrespective of the size of the problem. For the illustration in Fig. 3, the ~ 60 non-zero locations retained for $|(S_x C_x^T)_{(:,250)}|$ are depicted by the region between the red lines. For a problem with grid dimensions 500×500 this leads to 8 times fewer kept entries per row than for the unfiltered case of 500 non-zeros. On accounting for the Kronecker products as well as the identity nature of the block diagonal entries, the fully filtered $U_c U_c^T$ will have 128 times fewer non-zeros compared to the unfiltered one. The savings, of course, will only increase with problem size—for example, rows or columns of $U_c U_c^T$ for a grid of dimensions 2500×2500 will have the same decay rate as the 500×500 case, and thus the same number of nonzero entries to be stored.

We note that the above mentioned theoretical estimates of the sparsity pattern are based on two assumptions: (i) a constant diagonal matrix $I_s + \alpha\Lambda$, and (ii) an equal segregation of $U_c \equiv [F_u, F_v]^T$; *i.e.*, for the i^{th} column of U_c, the associated discrete Fourier functions $F_{u(i)}$ and $F_{v(i)}$ are afforded equal weighting so that $||F_{u(i)}||^2 = ||F_{v(i)}||^2 = 0.5$. Regarding assumption (i), $I_s + \alpha\Lambda$ is not a constant matrix but has a low condition number (as mentioned above), and therefore does not significantly alter the sparsity pattern of $U_c U_c^T$. Regarding assumption (ii), the non-equal weighting of the eigenvectors can be accounted for by incorporating diagonal matrices D_x and D_y that unequally scale the different discrete Fourier functions: $U_c = [F_u D_x, F_v D_y]^T$. This unequal weighting to the columns can be shown to only distribute the sparsity pattern across the diagonal and off-diagonal blocks, and not affect the overall number of non-zeros per row of $U_c U_c^T$.

Finally, returning to our original goal—we are interested in the overall sparsity of $B = E_0 C (C^T A C)^{-1} C^T E_0^T$ instead of $C(C^T A C)^{-1} C^T$ alone. We note that E_0 contains narrow delta functions (see Sec. 3.1) that are discrete analogues to the Dirac delta function, and is therefore sparse with only a few nonzero entries off of each diagonal. Additionally, since E_0 is a rectangular matrix, the overall size of $E_0 C (C^T A C)^{-1} C^T E_0^T$ is further reduced, allowing for further efficiency gains in storing B. For illustration, consider a sub-domain of grid dimensions 100×100 in the above-described 500×500 flow domain. One of the columns of B constructed for this sub-domain is plotted in Fig. 4. This column corresponds to the velocity induced on the sub-domain points due to a unit-valued surface stress in the x-direction applied at a grid point lying at the center of the sub-domain. The first and last 10^4 rows of B plotted in Fig. 4a and 4b correspond to this induced velocity in $x-$ and y-directions, respectively. The sharp peaks observed in these plots imply that the induced velocity due to a localized surface stress is concentrated around the grid point where the stress is applied. These plots therefore show that B is compact and can be approximated sparsely. For instance, based on the same relative drop tolerance of 10^{-2}, only the regions within the red contour lines in Fig. 4 need to be stored, thereby yielding significant storage benefits as compared to storing the full B matrix.

3.2.2. Drop tolerance filtering technique

We now describe the drop tolerance filtering technique to construct B sparsely. In this strategy, a drop tolerance parameter, ϵ, is used to filter out the elements of the matrix having relative magnitudes lower than the set tolerance. If we denote the sparsified version of B as B', then the filtering process is given as,

$$b'_{i,j} = \begin{cases} b_{ij} & \text{if } |b_{ij}| > \epsilon\, |b_{ii}| \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

where b_{ij} and b'_{ij} are the $(i,j)^{th}$ element of B and B', respectively. Hereby, B is replaced by the filtered matrix B' in our proposed sub-domain based IB method.

Since this filtering technique introduces additional approximations in the algorithm, the choice of ϵ should be made judiciously. A large choice of ϵ will proportionally filter out a large portion of B and result in an unstable or inaccurate algorithm. On the other hand, a small ϵ will yield only minimal storage gains. Through an ϵ-convergence study in Sec. 5.1.2, a drop tolerance of $\epsilon = 0.007$ is observed to strike the right balance between accuracy of the solutions and the storage requirements. This value of ϵ is shown to be suitable for a variety of problems described in Sec. 5.

Finally, we emphasize that, in practice, we do not construct the full matrix B before applying the filter. Instead, the columns of B are constructed one at a time by successively computing the action of B on a canonical unit vector as,

$$\left(E_0 C (C^T A C)^{-1} C^T E_0^T\right) e_j = B_j \tag{35}$$

where $e_j \in \mathbb{R}^{2n_{sd}}$ is the j^{th} canonical unit vector and B_j is the j^{th} column of B. The filter (34) is then applied on B_j before the next column, B_{j+1}, is evaluated. This construction process is conducive to scaling up for larger problem sizes.

3.3. Summary of the proposed sub-domain approach

To summarize, the time-varying IB interpolation operator $E_{n+1}^{(k)}$ defined on the moving Lagrangian body points is approximated via an interpolation of the time-independent IB interpolation operator E_0 defined on a fixed set of Eulerian sub-domain points. This allows us to precompute B and circumvent the expensive $(C^T A C)^{-1}$ solves traditionally required in Eq. (17). Furthermore, since B is compact, it is sparsified to achieve significant gains in storage requirement. In Appendix C, we show that the former sub-domain approximation, $P_{n+1}^{(k)} E_0$, converges to true $E_{n+1}^{(k)}$ as $\mathcal{O}(\Delta x^2)$, while the latter sparsity approximation is consistent in the sense that the residual of the surface stress equation with the sub-domain approximation (*i.e.* Eq. (23) substituted into Eq. (17)) converges to zero as $\epsilon \to 0$. The full fractional step algorithm from Eq. (16)-(20) for our proposed sub-domain approach can be now written as,

$$s^* = (C^T A C)^{-1} r_n^f \tag{36}$$

$$\left(P_{n+1}^{(k)} B' P_{n+1}^{(k)T} + \frac{2\Delta s}{\Delta t} S_{t,n+1}^{(k)T} J_t^{-1} S_{t,n+1}^{(k)} + \frac{2}{\Delta t} \hat{I}_d^T J_d^{(k)^{-1}} S_{d,n+1}^{(k)}\right) f_{n+1}^{(k+1)} =$$
$$P_{n+1}^{(k)} E_0 C s^* - r^{c(k)} + S_{t,n+1}^{(k)T} \left(r^{\theta(k)} - \frac{2}{\Delta t} J_t^{-1} r^{\phi(k)}\right) + \hat{I}_d^T \left(r^{\chi(k)} - \frac{2}{\Delta t} J_d^{(k)^{-1}} r^{\zeta(k)}\right) \tag{37}$$

$$\Delta\theta = J_t^{-1} \left(r^{\phi(k)} + \Delta s S_{t,n+1}^{(k)} f_{n+1}^{(k+1)}\right) \tag{38}$$

$$\Delta\chi_d = J_d^{(k)^{-1}} \left(r^{\zeta(k)} + S_{d,n+1}^{(k)} f_{n+1}^{(k+1)}\right) \tag{39}$$

$$s_{n+1} = s^* - (C^T A C)^{-1} C^T E_0^T P_{n+1}^{(k)T} f_{n+1} \tag{40}$$

Note that $E_{n+1}^{(k)}$ in Eq. (16)–(20) is replaced by $P_{n+1}^{(k)} E_0$ in Eq. (36)–(40) wherever applicable and the sparsified operator B' is used instead of B in Eq. (37). Similar to [26], the proposed approach in the vorticity-streamfunction formulation can be used in 3D without modification apart from the finite difference operators in going from 2D to 3D. We also derive the sub-domain based IB method that treats the fluid in primitive variables in Appendix D. This method utilizes the same sub-domain approximation, $E_{n+1}^{(k)} \approx P_n^{(k)} E_0$, and can be applied in both 2D and 3D.

The entire sub-domain based IB method can be divided into offline and online stages. The offline stage is only performed once at the beginning of the simulation to compute B'. In the online stage, the system of equations (36)-(40) is solved for the flow and structure variables and advanced in time. These stages are summarized in Algorithms 1 and 2, respectively.

We note that the offline stage involves performing $(C^T A C)^{-1}$ operations for every point in the sub-domain. Therefore, for a large and finely discretized sub-domain, precomputing B' can be an expensive process. However, we emphasize that it needs to be performed only once in the simulation. Furthermore, B' is independent of the instantaneous position of the bodies involved in the simulation. Therefore, B' constructed for a specific problem can be reused for several other problems provided that the following two conditions are met: (a) the spatial and temporal discretization sizes, Reynolds number and sub-domain remain unchanged and (b) all the bodies are guaranteed to stay within the sub-domain at all

Algorithm 1 Offline stage.

Input: Problem setup and grid
Output: Precomputed and sparsified matrix B'
1: Define a sub-domain according to the guidelines in Sec. 3.1.1
2: Construct E_0 using Eq. (25)
3: **for** $j \leftarrow 1$ to $2n_{sd}$ **do**
4: Compute j^{th} column B_j from Eq. (35)
5: Apply filtering: $B'_j \leftarrow filter(B_j)$ where $filter$ refers to the drop tolerance filtering technique in Eq. (34)
6: **end for**

Algorithm 2 Online stage.

Input: Initial conditions s_0, f_0 and χ_0; precomputed matrix B'
Output: s_n, f_n and χ_n for $n = 1, \ldots, t_{max}$
1: **for** $n \leftarrow 0$ to t_{max} **do**
2: Compute s^* from Eq. (36)
3: Initiate FSI iterations; $k \leftarrow 0$, $f_{n+1}^{(0)} = f_n$ and $\chi_{n+1}^{(0)} = \chi_n$
4: **while** $||\Delta\chi||_\infty > \varepsilon$ **do**
5: Choose sub-domain points for interpolation based on Sec. 3.1.3 and construct interpolation matrix $P_{n+1}^{(k)}$ with weights from Eq. (29).
6: Compute $P_{n+1}^{(k)} B' P_{n+1}^{(k)T}$ sparsely and other structural operators
7: Solve Eq. (37) via GMRES for $f_{n+1}^{(k+1)}$
8: Update position of the body $\chi_{n+1}^{(k+1)}$ via Eq. (38) and (39)
9: Advance FSI iterations $k \leftarrow k + 1$
10: **end while**
11: Compute s_{n+1} from Eq. (40)
12: **end for**

times. These conditions are conducive to parametric studies of flow problems, where only the body geometry or parameters such as mass ratio, stiffness *etc.* are varied without modifying the underlying discretization or sub-domain. Therefore, such parametric studies, which are customary in the fluid dynamics community, can be efficiently performed using our proposed sub-domain based IB method.

4. Parallel implementation

In this section, we describe the parallelization strategies implemented on the proposed sub-domain based IB approach to make it scalable across multiple CPUs.

4.1. Domain decomposition for fluid domain

Domain decomposition is a technique used in parallel computing where the computational domain is partitioned among many processors and each processor solves a part of the same system of equations locally. During these local computations, any required information from the neighboring processors is communicated via a communication protocol. In this work we use the message passing interface (MPI) protocol. Domain partitioning is performed using the Portable, Extensible Toolkit for Scientific Computation (PETSc) [27] which is built using the MPI library.

The Poisson like operations involving $(C^T A C)^{-1}$ are solved efficiently using fast sine transforms provided by the distributed-memory Fast Fourier Transform in the West (FFTW) MPI library [28]. FFTW MPI requires that the domain be partitioned in only one dimension irrespective of a two or three dimensional flow domain. In Fortran, this partitioning is done along the last dimension of the domain; for instance, the y-direction for 2D problems and the z-direction for 3D. Fig. 5 illustrates this domain partitioning procedure where the y-dimension is partitioned among three processors labeled as 0, 1 and 2. The blue lines in the flow domain denote the location of partitioning. Each processor handles the data computation involving the purple grid points in their respective domains. The inter-processor communication required while performing fast Fourier transforms is also managed by FFTW MPI.

As part of the domain partitioning technique, PETSc provides communication protocols conducive to the finite difference scheme used in our approach. Therefore, the inter-processor communications involved in operations such as C and C^T for computation at the grid points at the boundaries of the partitioned domains are efficiently handled by PETSc.

4.2. Partitioning of structure and flow sub-domain

Eq. (37) is solved for the surface stress vector $f \in \mathbb{R}^{n_f}$ and the parallelization of Eq. (37) depends on the parallelization of f. This vector consists of surface stresses in all coordinate directions for all the bodies involved in the simulation. In this work, we partition the entire surface stress vector f among a subset of available processors as equally as possible. We note that, since the number of degrees of freedom n_f is very small compared to the flow grid points, over-partitioning f among a large number of processors can sometimes create a communication overhead which can result in negative scaling. Therefore, the choice of the number of subset processors is problem dependent. For all the problems considered in Sec. 5,

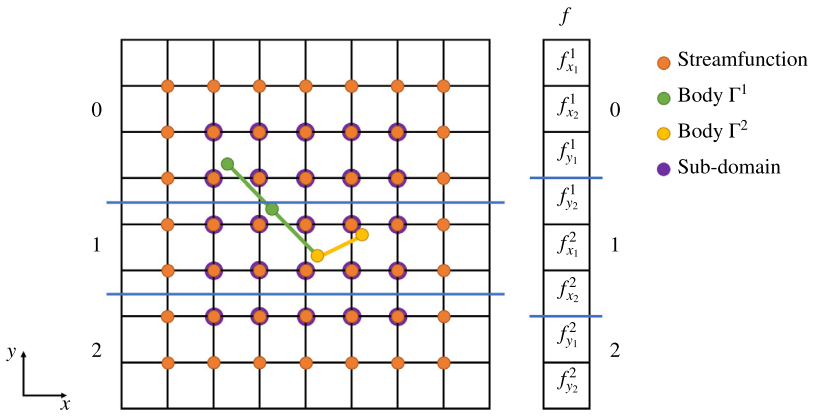


Fig. 5. Schematic of the domain decomposition of the fluid domain and partitioning of the surface stress vector among three processors labeled as 0, 1 and 2. The scripts in $f^i_{j_k}$ are, i: body number; j: $\{x, y\}$; k: body grid point. The blue lines denote the location of partitioning in the flow domain and surface stress vector. The schematic also describes the sub-domain grid points (purple markers) that coincide with the streamfunction/vorticity grid points (orange markers) on the cell vertices.

we partition f among all the processors since we did not observe the aforementioned overhead. The partitioning procedure of the surface stress vector is also illustrated in Fig. 5 where we consider a simple case of two bodies denoted by green and yellow points. We stack the surface stresses in the order of the number assigned to the body with stresses in x-direction stacked first followed by y-surface stress. This force vector is partitioned among three processors as denoted by the blue lines. Finally, Eq. (37) is solved in parallel using GMRES which is also provided by PETSc.

The sub-domain and related operators are also partitioned similarly to the surface stress. For instance, the sub-domain IB interpolation operator $E_0 \in \mathbb{R}^{2n_{sd} \times n_q}$ and the operator in Eq. (37) $B' \in \mathbb{R}^{2n_{sd} \times 2n_{sd}}$ are partitioned equally along the first dimension *i.e.* rows having a global dimension of $2n_{sd}$. The sparse interpolation operator $P^{(k)}_{n+1} \in \mathbb{R}^{n_f \times 2n_{sd}}$ is also partitioned along the first dimension, but having a dimension n_f and evaluated locally.

4.3. Parallel interfacing between fluid and structure

Although the above-mentioned flow domain and surface stress partitioning approaches ensure equal load-balancing across processors in their respective flow or structural domain, parallel interfacing between them is not trivial. For instance, consider the interpolation of velocity from the flow grid to the body points via Eq, where $q \equiv Cs$ is a generic velocity vector. Here, q in the flow domain and E of the body are partitioned via fundamentally different strategies. Therefore, to enable parallel interfacing, the velocity at flow points within the support of the delta function at the body point in consideration are "scattered" or communicated to the processor owning that body point. Once the scattering of the velocity data is performed, Eq can be trivially performed as a sparse matrix-vector multiplication.

The exact same strategy is used for performing $E_0 q$ on the sub-domain in Eq. (37). However, the size of $E_0 q$ is potentially much larger than Eq, $n_{sd} \gg n_f$. Therefore, to improve the computational efficiency of performing $E_0 q$, it is evaluated at only those vector locations where the corresponding column of $P^{(k)}_{n+1}$ is non-zero since we eventually only need to evaluate the overall matrix-vector product $P^{(k)}_{n+1} E_0 q$.

5. Results

In this section, we test the computational accuracy and efficiency of our proposed sub-domain based IB approach on several 2D FSI problems. Although our formulation in Sec. 2 is developed for FSI problems involving arbitrary number of rigid, torsionally mounted and deformable bodies, for simplicity, the 2D problems considered in this section consist of a combination of rigid and torsional bodies. The first problem consists of flapping of torsionally connected ellipses where we verify the accuracy of our sub-domain-based approach by comparing the results with those obtained by Wang and Eldredge [14] and using the true E, E^T operators (Eq. (16)–(20)) in place of the sub-domain interpolation approximations (Eq. (36)–(40)). In the second problem, the use of a compact sub-domain is demonstrated on flow around a stationary airfoil with a passively deployable flap. The computational efficiency of our sub-domain approach is compared with that attained when using the true E, E^T operators. These first two problems are constructed to highlight the accuracy and algorithmic efficiency of our proposed sub-domain-based interpolation approach. We then demonstrate the parallel scalability of our proposed method on a third problem consisting of 8 million grid points and increased complexity of a system of three airfoils in tandem each equipped with three passively deployable flaps.

A multi-domain approach for far-field Dirichlet boundary conditions of zero vorticity is incorporated for solving the flow equations where a hierarchy of grids of increasing coarseness stretching to the far field is employed (see reference [20]

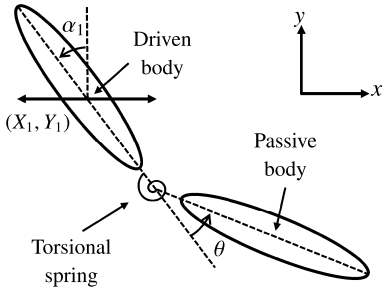


Fig. 6. Schematic of the flapping of two ellipses connected by a torsional spring.

for details). Following Goza and Colonius [15], the immersed boundary spacing is set to be twice as that of the flow grid spacing of the finest grid. A convergence criteria of $\|\Delta\theta\|_\infty \leq 10^{-7}$ is used when iterating between Eq. (37) and (38). The relative error used in various grid convergence and comparison studies in this section is defined as,

$$\text{Error}(\%) = \frac{||\eta - \eta_{ref}||_2}{||\eta_{ref}||_2} \times 100 \tag{41}$$

where η is the quantity of interest compared against a reference η_{ref}.

5.1. Flapping of torsionally connected ellipses

5.1.1. Problem description

This problem involves flapping of a 2D wing modeled in Wang and Eldredge [14]. The wing is modeled as two ellipses of chord length c having aspect ratios of 5:1, connected via a torsional spring. A schematic of this problem is shown in Fig. 6: a 'driven' component oscillates according to prescribed kinematics, and a second component that is hinged at one end of the driven body undergoes dynamics determined by the balance of aerodynamic and structural (stiffness and inertial) forces. The bodies are separated by a gap of width $0.1c$. The dimensional equation of motion for the hinge deflection angle θ between the bodies is given by,

$$I\ddot{\theta} + C\dot{\theta} + K\theta = M_f - [mL_2\cos(\alpha_1+\theta)]\ddot{X}_1 - [I_t^i + m(L_2^2 + L_1L_2\cos\theta)]\ddot{\alpha}_1 - mL_1L_2\sin\theta\dot{\alpha}_1^2 \tag{42}$$

where m is the mass of the passive ellipse, M_f is the dimensional moment due to the aerodynamic body forces analogous to the integral term in Eq. (3) and $L_1 = L_2 = 0.55c$ are the distances from the center of gravity of the respective ellipses to the hinge. The stiffness and damping coefficient of the spring are $K/(\rho_f f^2 c^4) = 456$ and $C/(\rho_f f c^4) = 3.95$, respectively, where f is the frequency of oscillations of the driven body. The moment of inertia of the second ellipse is $I/(\rho_f c^4) = 0.2886$ which is equivalent to a density ratio of $\rho_s/\rho_f = 5$. The multi-domain approach for far-field boundary conditions uses 5 grids of increasing coarseness where the finest and coarsest grid levels are $[-3.15, 3.15]c \times [-4.65, 1.65]c$ and $[-50.4, 50.4]c \times [-51.9, 48.9]c$, respectively.

The kinematics prescribed on the driven ellipse are same as that were used in Wang and Eldredge [14], given by

$$X_1(t) = \frac{A_0}{2}\frac{G_t(ft)}{\max G_t}C(ft) \tag{43}$$

$$Y_1(t) = 0 \tag{44}$$

$$\alpha_1(t) = -\beta\frac{G_r(ft)}{\max G_r} \tag{45}$$

where the translational and rotational shape functions, $G_t(t)$ and $G_r(t)$, respectively are given by,

$$G_t(t) = \int_t \tanh[\sigma_t\cos(2\pi t')]dt' \tag{46}$$

$$G_r(t) = \tanh[\sigma_r\cos(2\pi t)] \tag{47}$$

The initial impulsive velocity is avoided by using a start-up conditioner given by,

$$C(t) = \frac{\tanh(8t-2) + \tanh 2}{1 + \tanh 2} \tag{48}$$

Based on these kinematic parameters, the rotational Reynolds number is defined as,

Table 1
Kinematic parameters for the flow problem of flapping of torsionally connected ellipses.

Case No.	A_0/c	β	σ_t	σ_r	Re_r
1	1.4	$\pi/4$	3.770	3.770	100
2	1.4	$\pi/4$	0.628	0.628	100

Table 2
Parameters for grid convergence study and corresponding discrepancies in θ and c_l reported with respect to the finest case of $\Delta x/c = 0.00525$ for the problem of flapping of torsionally connected ellipses.

$\Delta x/c$	$\Delta t/\tau_r$	Discrepancy in θ	Discrepancy in c_l
0.00525	0.00163		
0.0105	0.00326	0.75%	3.18%
0.021	0.00652	2.75%	9.79%
0.042	0.01304	8.07%	17.95%

Table 3
Parameters for ϵ-convergence study and corresponding discrepancies in θ and c_l reported with respect to the finest case of $\epsilon = 0.000875$ for the problem of flapping of torsionally connected.

ϵ	Discrepancy in θ	Discrepancy in c_l	Memory (GB)
0.000875			20.02
0.00135	0.047%	0.39%	10.48
0.0035	0.088%	0.68%	5.39
0.007	0.13%	0.98%	2.73
0.014	0.17%	1.35%	1.35

$$Re_r = \frac{2\pi\beta\sigma_r}{\tanh\sigma_r}\frac{fc^2}{\nu} \tag{49}$$

We consider two test cases corresponding to the kinematic parameters provided in Table 1. See reference [29] for a detailed study of these parameters on the physics and aerodynamics of flapping.

To set the boundaries of the rectangular sub-domain for our proposed approach, firstly we determine the maximum limits of the body displacements. The maximum y-limits of the body displacements are $[-1.6, 0.5]c$ which may occur when $\alpha_1 = 0$ and $\theta = 0$. For the x-limits, although the maximum body displacements are $[-1.99, 1.99]c$ based on $\max(\alpha_1) = \beta = \pi/4$ and $\max(X_1) = A_0/2c = 0.7$, these maximum conditions never occur simultaneously since they are separated by a $\pi/2$ phase difference. Based on these conditions, the sub-domain is set to $[-1.89, 1.89]c \times [-1.66, 0.65]c$ which is a conservative estimate of the maximum limits of the body displacements. The sub-domain bounding the flapping ellipses is displayed as a black rectangular box in Fig. 8 for reference.

5.1.2. Implementation

Firstly, a grid convergence study on the first test case is performed by varying the spatial and temporal discretizations of the finest domain, $\Delta x/c$ and $\Delta t/\tau_r$, respectively, as shown in Table 2, where $\tau_r = (2\pi\beta\sigma_r f/\tanh\sigma_r)^{-1}$ is the characteristic rotation time. The discrepancy in the deflection angle, $\theta(t)$, and lift coefficient, $c_l(t) = 2F_y(t)/\rho_f^3 c^3$, in $0 < t/T < 3$ computed using Eq. (41) are used for determining convergence, where $T = f^{-1}$ is the time period and F_y is the total force on both ellipses in the y-direction. In this grid convergence study, the finest grid with $\Delta x/c = 0.00525$ is set to be the reference case against which the changes in deflection angle and lift are evaluated. Since the grid with $\Delta x/c = 0.0105$ is converged to within 1% of the finest grid for θ as shown in Table 2, $\Delta x/c = 0.0105$ and $\Delta t/\tau_r = 0.00326$ are used for presenting the results. Next, the order of spatial convergence p is determined via Richardson extrapolation as,

$$p = \log\left(\frac{|\eta_{r^2\Delta x} - \eta_{r\Delta x}|}{|\eta_{r\Delta x} - \eta_{\Delta x}|}\right)/\log(r) \tag{50}$$

where η is a flow metric evaluated for successively refined grids with a constant refinement ratio of r and subscript denotes the relative grid under consideration. In this problem, we set $\eta \equiv \theta(t)$ and $r = 2$. By using the first three grids in Table 2 and averaging p in $0 < t/T < 3$, we get the spatial order of accuracy to be $p = 1.53$. This is in agreement with the order of accuracy of most IB methods of between first and second order [3].

Next, an ϵ-convergence study on the first test case is performed by varying the drop tolerance parameter, ϵ, as shown in Table 3. Similar to the grid convergence study, the discrepancy in $\theta(t)$ and $c_l(t)$ in $0 < t/T < 1$ computed using Eq. (41) are tabulated. Furthermore, the memory requirements of storing the sparse B' matrix in gigabytes for the various ϵ values are provided. This memory includes the requirement of storing both the non-zero values and the column indices corresponding to those values of the sparse B'. It can be seen from Table 3 that the discrepancies in both θ and c_l across ϵ are minimal.

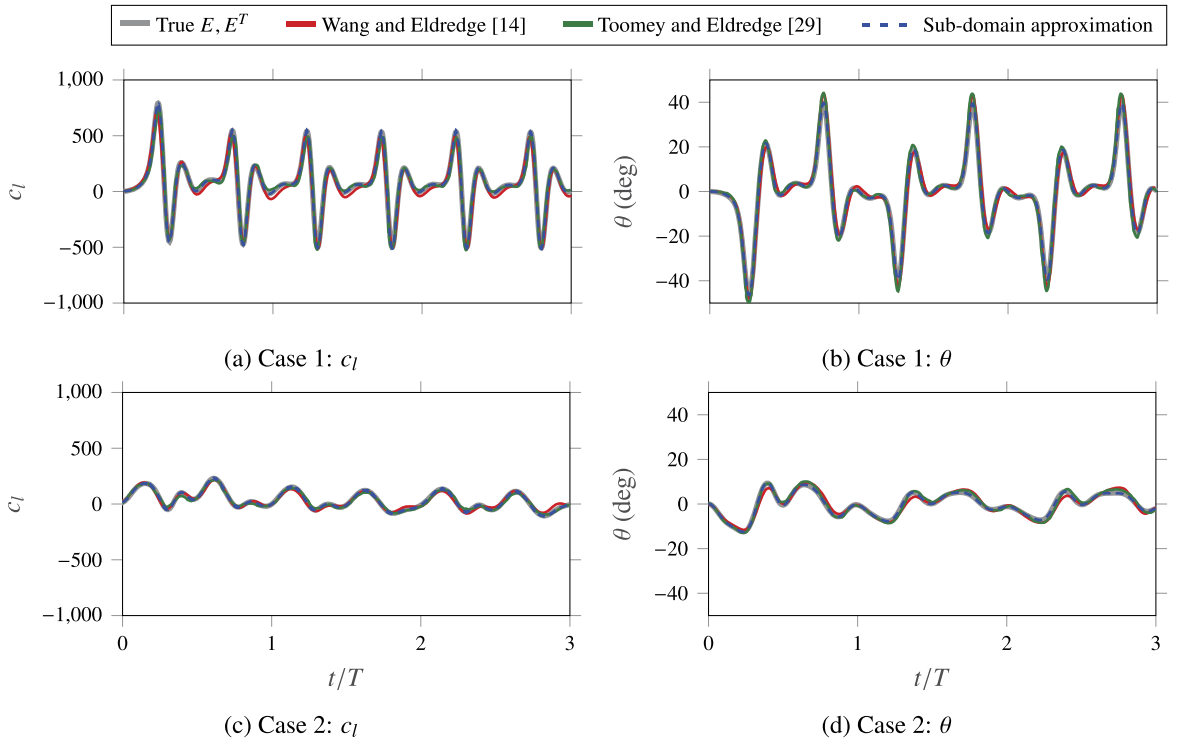


Fig. 7. Plots of lift coefficient, c_l and deflection angle, θ for the two cases obtained by the true E, E^T approach, Wang et al. [14], Toomey et al. [29] and our present approach for the problem of flapping of torsionally connected ellipses.

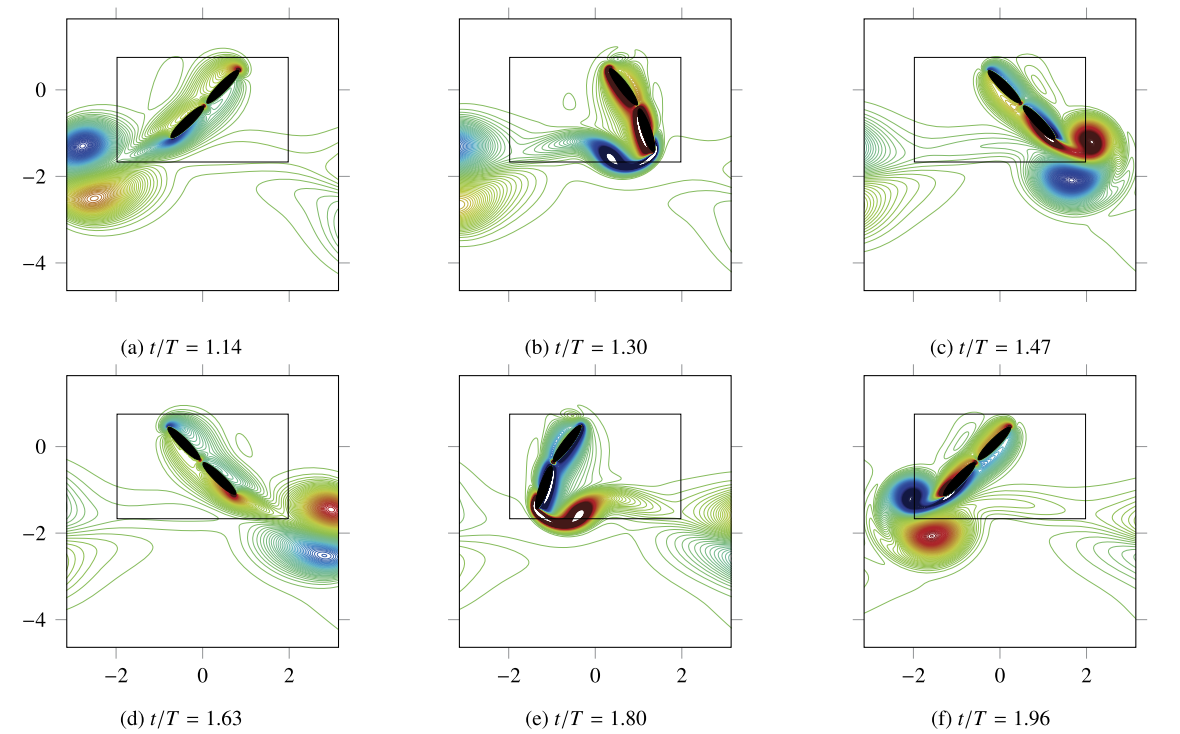


Fig. 8. Contour plots of vorticity at different time instants for case 1 of the problem of flapping of torsionally connected ellipses. The black rectangular box denotes the sub-domain that bounds all the bodies.

Table 4
Demonstration of computational accuracy via relative errors in θ and c_l, and speed-ups in using the proposed sub-domain approach compared to the true E, E^T operators for the problem of flapping of torsionally connected ellipses.

Case	Error in $\theta(t)$	Error in $c_l(t)$	Speed-up
1	0.20%	1.80%	18.39
2	0.88%	3.03%	12.77

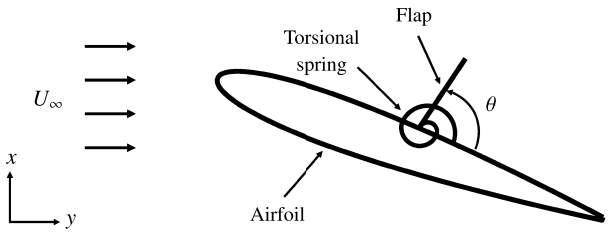


Fig. 9. Schematic of the system of passively deployable flap on an airfoil.

However, a significant rise in memory requirement is observed as ϵ is reduced. For $\epsilon = 0.007$, c_l is converged to within 1% of the finest ϵ considered. Also, the corresponding memory requirement of 2.73 GB is feasible for running the simulation on a single core which typically has 8 GB random access memory (RAM). Therefore, $\epsilon = 0.007$ is used for presenting all the results hereafter.

Next, we probe the accuracy of our proposed sub-domain approach by comparing the lift coefficient and deflection angle in Fig. 7, for the two test cases listed in Table 1, to those obtained by Wang and Eldredge [14], Toomey and Eldredge [29] and when using the true E, E^T in place of the sub-domain interpolation approximations. The temporal variations of the deflection angle and lift agree well across all four cases. For completeness, we illustrate the passive flapping of the second ellipse and the resulting lingering vortices via vorticity snapshots at different time instants in Fig. 8.

We provide the relative errors in the lift and deflection angle between our sub-domain interpolation approach and the use of the true E, E^T operators in Table 4. Relative errors of less than 1% and around 3% for the lift and deflection angle, respectively, are obtained, which are within the tolerance to which our results are converged; cf., Table 2. For all the cases considered, a maximum of three FSI iterations were required per time step. The computational efficiency of our approach is also demonstrated in Table 4 via the significant speed-up obtained by our sub-domain approach compared with use of the true E, E^T. Here, speed-up is defined as the ratio of mean wall-times incurred per time step in $0 < t/T < 1$ when the simulations are performed on a single core. The speed-up of more than an order of magnitude is due to the elimination of the bottleneck described in Sec. 2.3.

5.2. Passively deployed flap on an airfoil

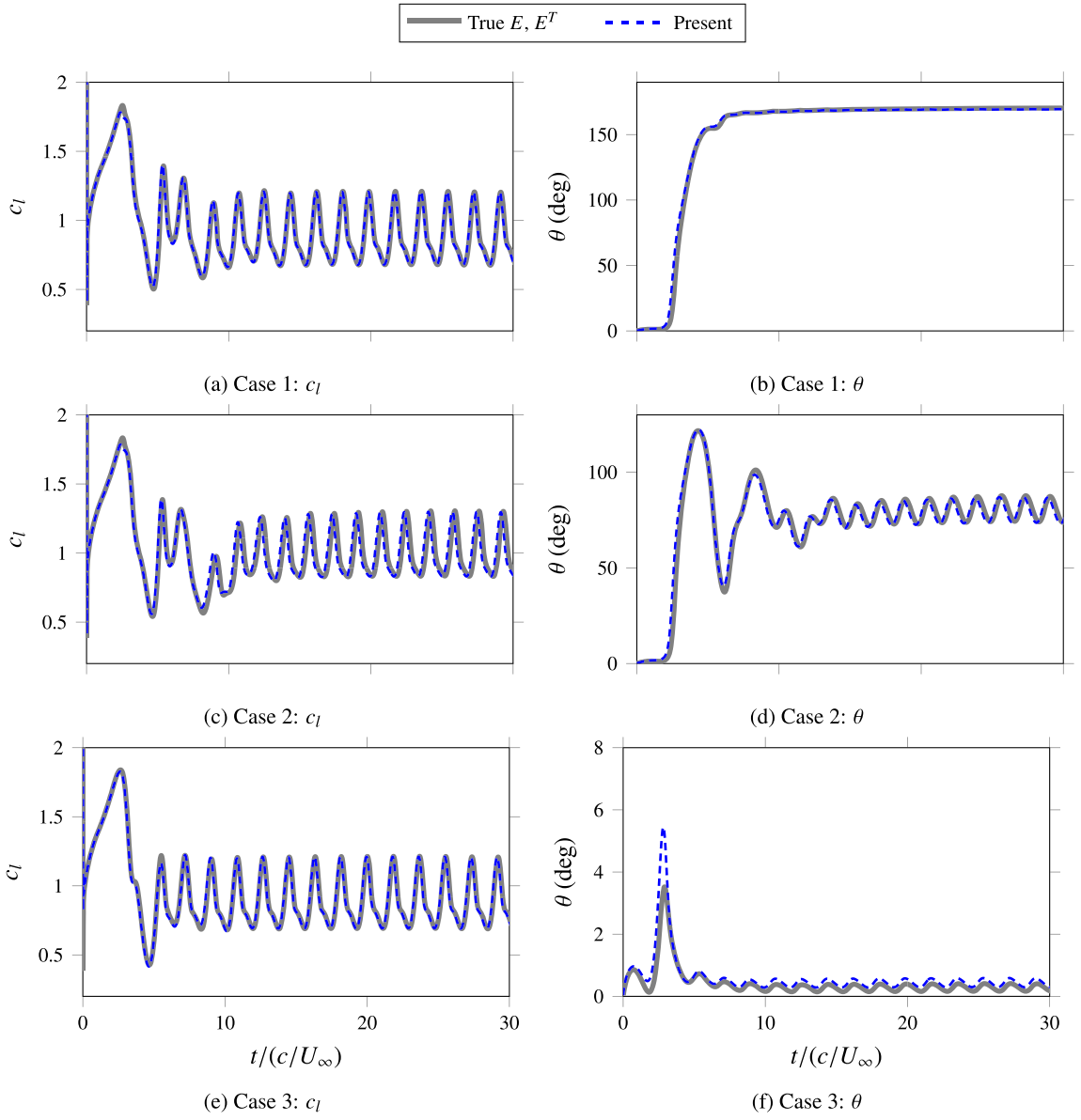
5.2.1. Problem description

This problem consists of a stationary NACA0012 airfoil of chord length c at an angle of attack of 20° in a flow with freestream velocity U_∞. The Reynolds number based on the chord length is set to 1000. A flap of length $0.2c$ is hinged on the upper surface of the airfoil at a distance of $0.5c$ from the leading edge via a torsional spring, as shown in Fig. 9. We fix the non-dimensional moment of inertia and damping coefficient to $i_t^i = I_t^i/\rho_f c^4 = 0.001$ and $c_t^i = C_t^i/\rho_f U_\infty c^3 = 0$, respectively and consider three test cases of widely varying stiffness, $k_t^i = K_t^i/\rho_f U_\infty^2 c^2 = \{0, 0.001, 0.1\}$. Initially, the flap is rested at an angle of 5° from the airfoil surface, which is taken as the undeformed (zero stress) deflection angle. As the vortex shedding process occurs, the flap passively deploys and interacts with the flow, providing significant lift improvements compared to the flap-less case [30,31]. For the multi-domain approach for far-field boundary conditions, five grids of increasing coarseness are used where the finest and coarsest grid levels are $[-0.5, 2.5]c \times [-1.5, 1.5]c$ and $[-23, 25]c \times [-24, 24]c$, respectively.

We chose the airfoil-flap problem to demonstrate the use of a compact sub-domain to reduce the storage requirements of the precomputed matrix B'. Since the airfoil is stationary and only the flap undergoes large displacements, we construct a small rectangular sub-domain that bounds only the physical limits of flap displacements. Accordingly, the rectangular sub-domain is set to $[0.23, 0.7]c \times [-0.24, 0.1]c$. This sub-domain bounding the flap motion is displayed as a black rectangular box in Fig. 11 for reference. Now, to account for the stationary airfoil, the exact airfoil body points are appended into the set of sub-domain points. These exact body points also allow us to use the exact IB interpolation operator E for the airfoil by setting the interpolation weight to one in P corresponding to the airfoil points. In problems such as these where physical knowledge of the problem is available that yield a compact sub-domain, significant savings in storing the precomputed matrix B' can be achieved. Finally, we emphasize that, since the underlying discretization sizes, sub-domain and Re are fixed, the precomputed matrix is only computed once for all the parametric variations considered within this test problem.

Table 5
Grid convergence test cases and corresponding errors in θ reported with respect to the finest case of $\Delta x/c = 0.0025$ for the problem of passively deployable flap on an airfoil.

$\Delta x/c$	$\Delta t/(c/U_\infty)$	Mean deflection $\bar{\theta}$	Discrepancy in θ (%)
0.0025	0.0003125	79.18°	
0.003	0.000375	79.60°	0.53
0.00349	0.0004375	79.96°	0.99
0.00395	0.0004935	77.93°	1.58
0.00455	0.000568	76.92°	2.85



(a) Case 1: c_l (b) Case 1: θ

(c) Case 2: c_l (d) Case 2: θ

(e) Case 3: c_l (f) Case 3: θ

Fig. 10. Plots of lift coefficient, c_l and deflection angle, θ for the three cases of stiffness, $k_t^i = \{0, 0.001, 0.1\}$, obtained by the true E, E^T approach and our present approach for the airfoil-flap system.

5.2.2. Implementation

Firstly, a grid convergence study on the test case of $k_t^i = 0.001$ is performed by varying the spatial and temporal discretizations of the finest domain as shown in Table 5. The mean deflection angle $\bar{\theta}$ in the limit cycle oscillation regime ($t/(c/U_\infty) > 20$) is used to determine grid convergence. In this grid convergence study, the finest grid with $\Delta x/c = 0.0025$ is set to be the reference case against which the relative changes of mean deflection angle are computed. Since the grid

Table 6
Results from the airfoil-flap problem: columns 2-3: demonstration of computational accuracy via relative errors in θ and c_l, and efficiency via speed-ups with respect to the true E, E^T operators; column 4: change in mean lift compared with a baseline case involving no flaps. The last column is not a measure of computational accuracy, but a demonstration of the potential aerodynamic benefits associated with torsionally-hinged flaps.

k_t^i	Error in $\bar{\theta}$	Error in $\bar{c}_l$	Speed-up	$\Delta\bar{c}_l$
0	0.47%	0.41%	4.22	-0.25%
0.001	0.83%	0.30%	3.87	14.55%
0.1	0.15°	0.48%	4.07	1.15%

with $\Delta x/c = 0.00349$ is converged to within 1% of the finest grid, $\Delta x/c = 0.00349$ and $\Delta t/(c/U_\infty) = 0.0004375$ are used for presenting the results.

Next, we determine the accuracy of our proposed sub-domain approach by comparing the lift coefficient and deflection angle in Fig. 10, for the various cases of stiffness, $k_t^i = \{0, 0.001, 0.1\}$, to those obtained when using the true E, E^T operators in place of the sub-domain interpolation approximations. Here, the lift coefficient is defined as $c_l = 2F_y/\rho_f U_\infty^2 c$ where F_y is the total force on the airfoil and flap system in the y-direction. It can be seen that the transient dynamics of the flap deploying into the flow (implied from the large initial deflection angles) and subsequent limit cycle oscillations produced from our sub-domain approach agree well with those obtained by using true E, E^T for all the cases. The plots of deflection angle also demonstrate the stability of our approach in the presence of large deflections for very low stiffness of $k_t^i = 0$ and $k_t^i = 0.001$. We note that the discrepancy in the deflection angle between our approach and that of true E, E^T for the largest stiffness case of $k_t^i = 0.1$ is slightly larger than the lower stiffness cases. This is because the flap oscillates very close to the airfoil resulting in the subset of the n_p nearest neighboring sub-domain interpolation points to be even closer to the airfoil for the immersed body points near the hinge. These sub-domain points unphysically include the contribution of the fictitious fluid within the airfoil towards the interpolation and regularization operations of the immersed boundary method. This effect is not prominent for the lower stiffness cases where the flap deflection angle is large. The problem associated with the fictitious fluid could be addressed by using the method of immersed layers [32] where Heaviside functions are used for distinguishing the physical and fictitious fluid regions. For all the cases considered, a maximum of only two FSI iterations were required per time step. The relative errors in the mean lift coefficient and deflection angle between our approach and the use of true E, E^T are also provided in Table 6. For all the cases, relative errors of less than 1% for both the lift and deflection angle are attained. Note that, for the case of $k_t^i = 0.1$, we have reported the absolute error in the mean deflection angle instead of the relative error because the flap oscillates very close to the airfoil with a mean deflection angle of 0.44° and 0.29° obtained from our sub-domain approach and by using true E, E^T, respectively. This results in a misleadingly high relative error of 52.79% with respect to such a small mean deflection angle while noting that the relative error in c_l is still below 1%.

The computational efficiency of our approach is demonstrated in Table 6 by reporting the speed-up attained by our proposed approach compared to when the true E, E^T operators are utilized. Here, the speed-up is defined as the ratio of mean wall-times on a single core incurred per time step over the first 1000 time steps ($t/(c/U_\infty) < 0.4375$). Our proposed sub-domain approach is approximately four times more efficient than when using the true E, E^T operators for this airfoil-flap problem.

Finally, to indicate the potential engineering utility of these deployable flaps in improving aerodynamic performance, we show in Table 6 the relative change in the lift coefficient, $\Delta\bar{c}_l$, for the airfoil-flap system compared with the flap-less case of only the airfoil at the same angle of attack and Re. The case with $k_t^i = 0.001$ provides significant lift benefits of around 15%. To understand the physical mechanisms that enable this lift improvement, four snapshots of the pressure field over one period of the limit cycle oscillation regime ($t/(c/U_\infty) > 20$) are plotted in Fig. 11. We can clearly observe a low pressure region denoted by blue color just upstream of the flap in all the contours. This low pressure zone is formed due to the trapping of a portion of the leading edge vortex by the flap. This low pressure region therefore augments the lift of the airfoil-flap system compared to the case without the flap. Similar physical mechanisms that augment lift have been found for statically deployed flaps [33], but to our knowledge this mechanism has not been observed for the case of dynamic flaps mounted via torsional springs. The lift variations for $k_t^i = 0$ and $k_t^i = 0.1$ are not significant since they either excessively or barely deploy the flap, respectively, such that the trapping of the vortex is not realized.

5.3. Airfoils with passively deployed flaps in tandem

5.3.1. Problem description

In this section, we demonstrate the parallel scalability of our proposed approach on a relatively large problem consisting of 8 million flow grid points. This problem involves a similar airfoil-flap system as described in the previous problem in Sec. 5.2, but with three stationary NACA0012 airfoils in tandem, each equipped with three torsionally hinged flaps. Such a tandem-airfoil-flap system is found to reduce the total drag coefficient compared to the tandem-airfoil system without any flaps (see the next Sec. 5.3.2 for details).

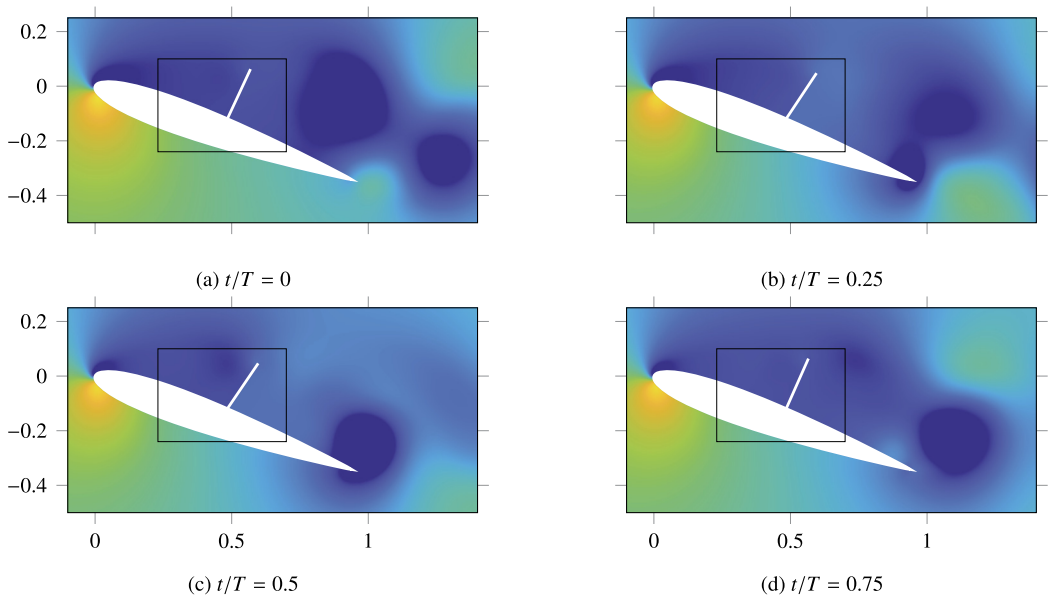


Fig. 11. Contour plots of pressure at four time instants in one time period T. Blue and yellow color denotes regions of low and high pressure, respectively. The black rectangular box denotes the sub-domain that bounds the flap motion.

The airfoils are separated by a distance of $1.18c$ between the consecutive leading edges where c denotes the chord length of the airfoils. The flaps are located at a distance of $0.25c$, $0.5c$ and $0.75c$ from the leading edge of their respective airfoils. The angle of attack of all the airfoils is 20° and Reynolds number of the flow based on c is set to 1000. The parameters for all the springs and flaps are $k_t^i = 0.001$, $c_t^i = 0$ and $i_t^i = 0.001$. Initially, all the flaps are rested at an angle of 5° from their respective airfoil tangential surface. As the vortex shedding process occurs, all flaps are allowed to passively respond to the aerodynamic forces.

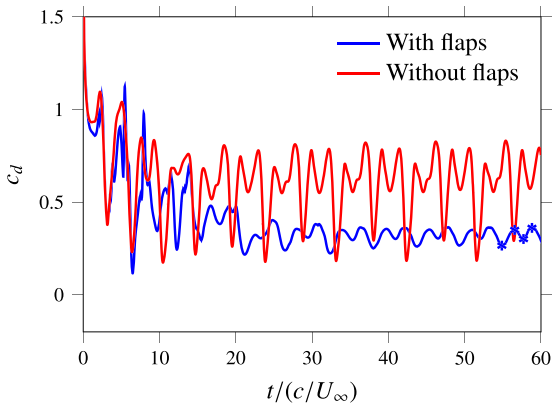
The multi-domain approach for the far-field boundary conditions employs five grids of increasing coarseness where the finest and coarsest grid levels are $[-0.5, 7.5]c \times [-2, 2]c$ and $[-60.5, 67.5]c \times [-32, 32]c$, respectively. The sub-domain boundaries are set to $[-0.008, 3.296]c \times [-0.348, 0.208]c$, which encompasses all the airfoils and the physical limits of flap displacements. This sub-domain bounding all the bodies is displayed as a black rectangular box in Fig. 13 for reference. The grid spacing of the finest domain is $\Delta x/c = 0.002$ and the time step size is $\Delta t/(c/U_\infty) = 0.00025$. Note that these discretizations are finer than those considered for the similar airfoil-flap problem considered in the previous section 5.2; therefore, a grid convergence study is not performed for this problem. The resulting size of the flow domain is 4000×2000, or 8 million grid points.

Recall from Sec. 4 that, for parallel implementation, the FFTW-MPI library requires that the domain decomposition of the flow domain be performed along the y-direction for 2D problems. For the tandem-airfoil-flap problem, this domain decomposition corresponds to 1D partitioning along the y-direction consisting of 2000 grid points. However, the preferred domain partitioning is along the x-direction, which has the larger dimension of 4000 grid points. We thus superficially rotate the original computational domain by 90° in clockwise direction to obtain a domain of 2000×4000 points. When displaying the results, the flow-fields are rotated back to the original 4000×2000 configuration for readability.

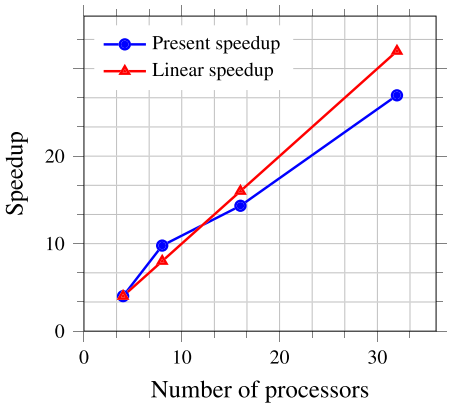
5.3.2. Implementation

First, the total drag coefficient of the tandem-airfoil-flap system, $c_d = 2F_x/\rho_f U_\infty^2 c$, where F_x is the total force on all airfoils and flaps in the x-direction, is plotted in Fig. 12a and compared with the case of the same three airfoils in tandem, but without any flaps. A reduction in mean drag, $\bar{c}_d$, by 46.92% is observed with respect to the flap-less case, where the mean is evaluated in the limit cycle oscillation regime after initial transients have decayed, $t/(c/U_\infty) > 30$. To indicate the physical mechanisms that enable this drag reduction, four snapshots of vorticity are plotted in Fig. 13. These snapshots correspond to two troughs and two peaks of one drag cycle in the limit cycle oscillation regime, indicated by the blue markers on the c_d plot in Fig. 12a. From these figures, we observe that the deployed flaps manipulate the flow to curve around a large "imaginary body" that acts as a streamlined connection of the true tandem-airfoil-flap system. Although significant flow separation occurs at the first airfoil, the leading flap deflects the shear layer in the upwards transverse direction, shielding the second and much of the third airfoil from drag-producing vortex interactions. The end result is a net reduction of drag for the collective system.

Now, we demonstrate favorable strong scaling by evaluating the speedup obtained over the first 1000 time steps $(t/(c/U_\infty) < 0.25)$ while increasing the number of processors as $\{4, 8, 16, 32\}$. Typically, the speedup is defined as the



(a) Plot of c_d v/s time for the tandem-airfoil systems with and without flaps.



(b) Plot of observed and linear speed-up v/s number of processors.

Fig. 12. Comparison of total drag coefficient, c_d (left) and demonstration of favorable strong scaling (right).

ratio of the time taken by one processor to that of p parallel processors. However, due to the large size of the problem, we instead define the speedup with respect to four processors as,

$$\text{Speedup} = \frac{T_4}{T_p} \times 4 \tag{51}$$

where T_p is the time taken by p processors. Note that we only take into account the time incurred in the online stage of our algorithm for calculating speedup. The scaling results are displayed in Fig. 12b by plotting the speedup versus the number of processors. A plot of linear (ideal) speedup is also provided for reference. A favorable strong scaling efficiency of 84.22% at $p = 32$ processors is observed where efficiency is defined as the ratio of speedup to p. While an ideal 100% efficiency at $p = 32$ is not achieved as compared to some highly scalable IB methods [34,35], the present speedup is comparable to that of other IB methods that scale favorably [36–39]. The strong scaling efficiency is highly dependent on the solver used to solve the high-dimensional fluid equations. In this work, we have utilized a highly efficient fast Fourier transform-based linear solver using the FFTW library whose computational complexity is only $N \log(N)$. Unfortunately, as mentioned in Sec. 4, the distributed FFTW MPI library requires that the domain be partitioned in only one dimension irrespective of a two- or three-dimensional flow domain. It is well-known in parallel programming that 1D domain decomposition exhibits reduced scalability as compared to multi-dimensional domain decomposition. Another drawback of FFTW MPI is that, it does not scale ideally [40] since it requires multiple matrix transpose operations for computing a single transform [28], which is inefficient in parallel. Furthermore, we also employ the multi-domain approach of Colonius and Taira [20] to efficiently incorporate far-field boundary conditions while utilizing uniform grids required for performing Fourier transforms. Unfortunately, the switching between grid levels in the multi-domain approach requires significant communication of data across processors before any computation can be performed in that grid level. This inability to overlap communication and computation also hampers the parallel efficiency of our IB method. In future efforts, we will look into new solution strategies of the spatially discrete governing equations, leveraging for example quadtree data structures for fast solutions on non-uniform grids, which could remove the multi-domain bottleneck altogether and also use solvers that allow multi-dimensional domain decomposition. Finally, for all the cases considered, a maximum of only two FSI iterations were required per time step.

6. Conclusions

In this manuscript, we have proposed an efficient sub-domain based IB approach that addresses the computational bottleneck encountered in a number of strongly and semi-strongly coupled IB methods, wherein several costly large dimensional systems are solved only for a small number of body variables. In our proposed approach, the fluid-structure coupling operator is constructed on a fixed set of flow sub-domain points instead of time-varying body points, allowing us to precompute a matrix that embeds the large dimensional system before any time advancement is performed. This precomputation process results in all FSI iterations being restricted to small-dimensional systems. As such, the proposed algorithm mimics favorable features of stationary-body IB methods, where the matrix that encodes the interface coupling can be precomputed, while retaining the desirable stability properties of strongly coupled FSI methods. We also formulated a parallel implementation of this sub-domain-based IB algorithm, and demonstrated favorable strong scaling.

Numerical experiments consisted of two dimensional flow problems involving large body displacements such as flapping of torsionally connected ellipses and the FSI dynamics of a passively deployable flap on an airfoil. The results obtained from our approach agreed well with those from the previous studies. Regarding computational efficiency, our approach

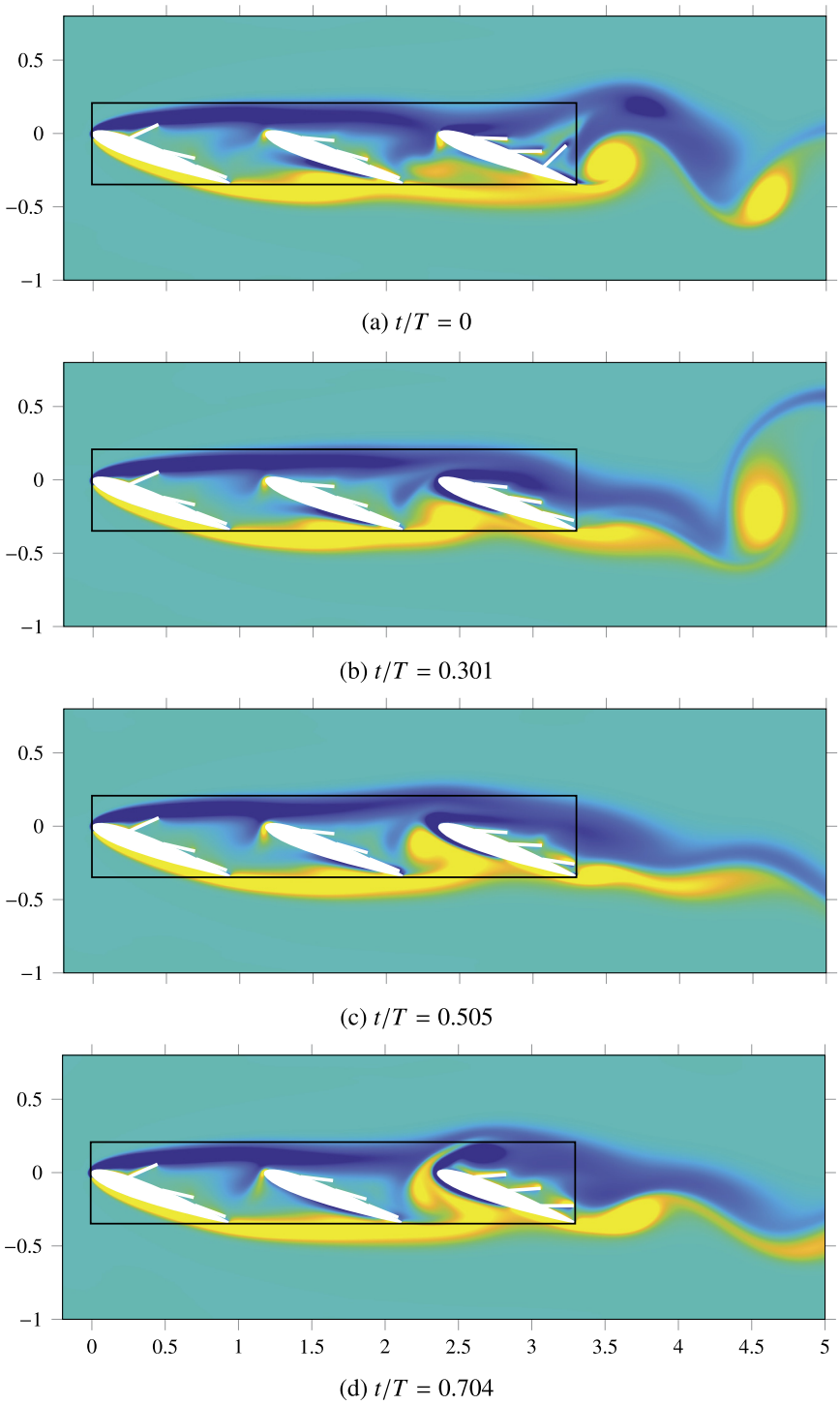


Fig. 13. Contour plots of vorticity at different time instants corresponding to the blue markers in the c_d plot of Fig. 12a. Blue and yellow denote regions of counter-clockwise and clockwise vorticity. Here T denotes the time period of limit cycle oscillations. The black rectangular box denotes the sub-domain that bounds all the bodies.

outperformed an implementation of the IB method without the proposed sub-domain approach, delivering speed-ups of up to an order of magnitude for the presented problems. Finally, favorable strong scaling of our parallel implementation was demonstrated on a larger problem consisting of three airfoils in tandem, each equipped with three passively deployable flaps. For all the cases considered, our approach produced a convergent solution in less than three FSI iterations.

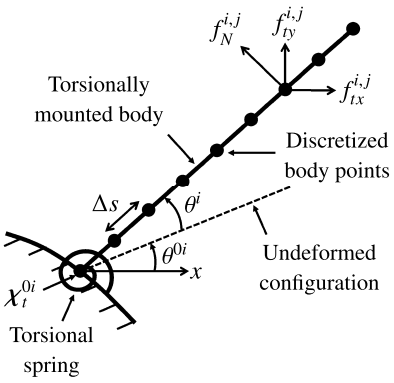


Fig. 14. Schematic of a torsionally mounted body.

In this manuscript, we have developed our sub-domain based IB method on the foundation of the IB method of Goza and Colonius [15]. However, we emphasize that our formulation can be extended to a wide range of strongly coupled IB methods, provided that these methods are able to be reformulated to restrict the FSI iterations to nominally small-dimensional systems. The proposed approach can also be expected to provide moderate speedups for non-FSI problems involving bodies with fully prescribed kinematics. Furthermore, we note that although the flow problems considered in this work consisted of a combination of rigid and torsional bodies, the formulation was developed and equally applicable for a more general setting that includes deformable bodies, possibly combined to create more complex structures.

Finally, we note that, in this work, we considered a simple rectangular sub-domain whose associated fluid-structure coupling matrix had to be precomputed only once. For future work, an adaptive algorithm can be developed wherein a new but compact and potentially non-rectangular sub-domain is constructed at specific time intervals. This adaptive approach, if optimized for the time intervals of reconstructing the sub-domain, can mitigate the storage requirements of the precomputed matrix without significantly compromising the computational speedups. Another avenue for future work involves utilizing the integrating factor approach of Liska and Colonius [41] where the columns of B can be constructed via convolution operations of lattice Green's function (LGF) with the sub-domain points. This could enable an efficient solution procedure to the discretized differential-algebraic system of equations for the surface stress as well as facilitate a relatively straightforward analysis of compactness of B.

CRediT authorship contribution statement

Nirmal J. Nair: Conceptualization, Methodology, Software, Formal analysis, Data Curation, Writing (Original and Revision). **Andres Goza:** Conceptualization, Methodology, Writing (Original and Revision), Supervision, Funding acquisition

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We gratefully acknowledge funding through the National Science Foundation under grant CBET 20-29028. The code for the proposed sub-domain based IB approach used to simulate the problems in this work is open-source and publicly available at https://github-dev.cs.illinois.edu/NUFgroup/IB_parallel.

Appendix A. Derivation of Eq. (16)–(20)

This appendix provides the derivation of the fully discretized and block LU factorized equations (16)–(20) from the governing equations (1)–(7). Firstly, the spatially discretized equations of motion for the fluid on a staggered uniform Cartesian grid in the vorticity-streamfunction formulation [20] are given by,

$$C^T C\dot{s} + \mathcal{N}(Cs) = C^T LCs - C^T E^T(\chi) f \tag{A.1}$$

where $\mathcal{N}(\cdot)$ is the discretization of the nonlinear advection term.

For the spatial discretization of the equation for the torsionally connected bodies, consider the schematic of the i^{th} torsional body Γ_t^i with an undeformed (zero stress) angle θ^{0i} from the x-axis in Fig. 14. The normal surface stress, f_N^i, exerted on the body by the fluid is given by,

$$f_N^i = -f_{tx}^i \sin(\theta^{0i} + \theta^i) + f_{ty}^i \cos(\theta^{0i} + \theta^i) = R_t^i f_t^i \tag{A.2}$$

where f_{tx}^i and f_{ty}^i are the surface stresses in the x and y directions, respectively; $f_t^i = [f_{tx}^i, f_{ty}^i]^T$; and R_t^i is a matrix containing two blocks of diagonal matrices aligned column-wise with diagonal entries $-\sin(\theta^{0i} + \theta^i)$ and $\cos(\theta^{0i} + \theta^i)$ corresponding to f_{tx}^i and f_{ty}^i, respectively. Accordingly, the moment due to surface stress can be discretized as,

$$-\int_{\Gamma_t^i} (\boldsymbol{\chi}_t^i - \boldsymbol{\chi}_t^{0i}) \times \boldsymbol{f}(\boldsymbol{\chi}_t^i) d\boldsymbol{\chi}_t^i \xrightarrow{\text{discretize}} \sum_{j=0}^{n_t^i} (j\Delta s)(R_t^i f_t^i)_j \Delta s = Q_t^i R_t^i f_t^i \Delta s \tag{A.3}$$

where n_t^i is the number of discretized points on Γ_t^i and $Q_t^i = [0, 1, \ldots, n_t^i]\Delta s$. Now, the semi-discretized equations for the torsional body are given by,

$$i_t^i \dot{\phi}^i + c_t^i \phi^i + k_t^i \theta^i = Q_t^i R_t^i f_t^i \Delta s + g_t^i \qquad \text{for} \quad i = 1, \ldots, m_t \tag{A.4}$$

where we define $\phi^i = \dot{\theta}^i$.

The equation for a deformable body is discretized using a finite element procedure as described in Goza and Colonius [15]. By expressing the structural variables using a set of compatible shape functions, we write the spatially discretized form of Eq. (4) as,

$$M_d^i \dot{\zeta}_d^i + R_d^i(\chi_d^i) = Q_d^i (g_d^i + W_d^i(\chi_d^i) f_d^i) \qquad \text{for} \quad i = 1, \ldots, m_t \tag{A.5}$$

where $f_d^i = [f_{dx}^i, f_{dy}^i]^T$ and the specific forms of M_d^i, R_d^i, Q_d^i and W_d^i containing the shape functions are described in reference [15]. Next, the boundary conditions on all the bodies are discretized as,

$$E_r^i C s = u_r^i \qquad \text{for} \quad i = 1, \ldots, m_r \tag{A.6}$$

$$E_t^i C s - R_t^{iT} Q_t^{iT} \phi^i = 0 \qquad \text{for} \quad i = 1, \ldots, m_t \tag{A.7}$$

$$E_d^i C s - \zeta_d^i = 0 \qquad \text{for} \quad i = 1, \ldots, m_d \tag{A.8}$$

Following the time discretization schemes of Goza and Colonius [15], the fully discretized equations are written as,

$$C^T A C s_{n+1} + C^T E_{n+1}^T f_{n+1} = r_n^f \tag{A.9}$$

$$\frac{4}{\Delta t^2} i_t^i \theta_{n+1}^i + \frac{2}{\Delta t} c_t^i \theta_{n+1}^i + k_t^i \theta_{n+1}^i - Q_t^i R_{t,n+1}^i f_{t,n+1}^i \Delta s = r_n^{\phi,i} \qquad \text{for} \quad i = 1, \ldots, m_t \tag{A.10}$$

$$\frac{2}{\Delta t} \theta_{n+1}^i - \phi_{n+1}^i = r_n^{\theta,i} \qquad \text{for} \quad i = 1, \ldots, m_t \tag{A.11}$$

$$\frac{4}{\Delta t^2} M_d^i \chi_{d,n+1}^i + R_d^i(\chi_{d,n+1}^i) - Q_d^i W_{d,n+1}^i f_{d,n+1}^i = r_n^{\zeta,i} \qquad \text{for} \quad i = 1, \ldots, m_d \tag{A.12}$$

$$\frac{2}{\Delta t} \chi_{d,n+1}^i - \zeta_{d,n+1}^i = r_n^{\chi,i} \qquad \text{for} \quad i = 1, \ldots, m_d \tag{A.13}$$

$$E_{r,n+1}^i C s_{n+1} = u_{r,n+1}^i \qquad \text{for} \quad i = 1, \ldots, m_r \tag{A.14}$$

$$E_{t,n+1}^i C s_{n+1} - R_{t,n+1}^{iT} Q_t^{iT} \phi_{n+1}^i = 0 \qquad \text{for} \quad i = 1, \ldots, m_t \tag{A.15}$$

$$E_{d,n+1}^i C s_{n+1} - \zeta_{d,n+1}^i = 0 \qquad \text{for} \quad i = 1, \ldots, m_d \tag{A.16}$$

where $r_n^f = (\frac{1}{\Delta t} C^T C + \frac{1}{2} C^T L C) s_n + \frac{3}{2} C^T \mathcal{N}(C s_n) - \frac{1}{2} C^T \mathcal{N}(C s_{n-1})$, $r_n^{\phi,i} = i_t^i \left(\frac{4}{\Delta t^2} \theta_n^i + \frac{4}{\Delta t} \phi_n^i + \dot{\phi}_n^i \right) + c_t^i \left(\frac{2}{\Delta t} \theta_n^i + \phi_n^i \right) + g_t^i$, $r_n^{\theta,i} = \phi_n^i + \frac{2}{\Delta t} \theta_n^i$, $r_n^{\zeta,i} = M_d^i \left(\frac{4}{\Delta t^2} \chi_{d,n}^i + \frac{4}{\Delta t} \zeta_{d,n}^i + \dot{\zeta}_{d,n}^i \right) + Q_d^i g_d^i$ and $r_n^{\chi,i} = \zeta_{d,n}^i + \frac{2}{\Delta t} \chi_{d,n}^i$. Following Goza and Colonius [15], an iterative procedure is introduced to solve the above system of equations. A guess at iteration (k) is used to compute a new guess at $k+1$ by defining, $\psi_{n+1}^{i(k+1)} = \psi_{n+1}^{i(k)} + \Delta\psi^i$ where $\psi = \{\theta, \phi, \chi_d, \zeta_d\}$ and $\Delta\psi^i$ is assumed to be small. On substituting this decomposition into (A.9)-(A.16) and retaining first order terms in the increments and Δt, we get,

$$\begin{bmatrix} C^T A C & 0 & 0 & 0 & 0 & C^T E_{n+1}^{(k)T} \\ 0 & 0 & J_t & 0 & 0 & S_{t,n+1}^{(k)} \Delta s \\ 0 & -I_t & \frac{2}{\Delta t} I_t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & J_d^{(k)} & S_{d,n+1}^{(k)} \\ 0 & 0 & 0 & -I_d & \frac{2}{\Delta t} I_d & 0 \\ E_{n+1}^{(k)} C & S_{t,n+1}^{(k)T} & 0 & \hat{I}_d^T & 0 & 0 \end{bmatrix} \begin{bmatrix} s_{n+1} \\ \Delta\phi \\ \Delta\theta \\ \Delta\zeta_d \\ \Delta\chi_d \\ f_{n+1} \end{bmatrix} = \begin{bmatrix} r_n^f + \mathcal{O}(\Delta t) \\ r_n^\phi - J_t \theta_{n+1}^{(k)} + \mathcal{O}(\Delta t) \\ r_n^\theta - \frac{2}{\Delta t} \theta_{n+1}^{(k)} + \phi_{n+1}^{(k)} \\ r_n^\zeta - J_d^{(k)} \chi_{d,n+1}^{(k)} + \mathcal{O}(\Delta t) \\ r_n^\chi - \frac{2}{\Delta t} \chi_{d,n+1}^{(k)} + \zeta_{d,n+1}^{(k)} \\ U_{b,n+1}^{(k)T} + \mathcal{O}(\Delta t) \end{bmatrix} := \begin{bmatrix} r_n^f \\ r^{\phi(k)} \\ r^{\theta(k)} \\ r^{\zeta(k)} \\ r^{\chi(k)} \\ r^{c(k)} \end{bmatrix} \tag{A.17}$$

Here, we have aggregated all the individual $\Delta\psi^i$ into a vector $\Delta\psi$ where $\psi = \{\theta, \phi, \chi_d, \zeta_d\}$ and for the right-hand side terms. I_t and I_d are identity operators of compatible sizes for the torsional and deformable bodies, respectively; J_t is a square diagonal operator of size m_t with diagonal elements $J_{t(i,i)} = \frac{4}{\Delta t^2} i_t^i + \frac{2}{\Delta t} c_t^i + k_t^i$; and $J_d^{(k)}$ is a square block diagonal operator having m_d blocks where the i^{th} diagonal block is given by $J_{d(i,i)}^{(k)} = \frac{4}{\Delta t^2} M_d^i + K_d^{i(k)}$ where $K_d^{i(k)} = dR_d^i/\chi|_{\chi=\chi_{d,n+1}^{i(k)}}$. The remaining operators are defined as $S_{t,n+1}^{(k)} = [0,\ -Q_t^{(k)} R_{t,n+1}^{(k)} \Delta s,\ 0]$, $S_{d,n+1}^{(k)} = [0,\ 0,\ -Q_d^{(k)} W_{d,n+1}^{(k)}]$, $\hat{I}_d = [0,\ 0,\ I_d]$, $U_{b,n+1}^{(k)} = [u_{r,n+1},\ R_{t,n+1}^{(k)T} Q_t^{(k)} \phi_{n+1}^{(k)},\ \zeta_{d,n+1}^{(k)}]$, where R_t, Q_t, W_d and Q_d are block diagonal operators with entries R_t^i, Q_t^i, W_d^i and Q_d^i, respectively. On performing a block LU decomposition of Eq. (A.17), we get the final system of equations given in Eq. (16)–(20).

Appendix B. Properties of interpolated delta functions

In immersed boundary methods, the delta function used to construct E_n in Eq. (24) is chosen such that it satisfies certain moment conditions, for instance,

$$\sum_j (\xi_i - x_j)^q d(\xi_i - x_j) = \begin{cases} 1, & q = 0 \\ 0, & q = 1 \\ K, & q = 2 \\ 0, & q = 3 \end{cases} \quad \text{for some constant } K \tag{B.1}$$

In this section we show that the interpolated delta functions that form the elements of the sub-domain-approximated operator $P_n E_0$ in Eq. (27) satisfy the moment conditions provided that the underlying delta function being interpolated, $d(\cdot)$, also satisfies these conditions. To simplify the analysis, we consider the delta function in x-direction only since the delta function in 2D is simply the outer product of delta functions in both directions. Accordingly, in the following 1D analysis, we use $n_p = 2$.

We begin the proof by rewriting the left hand side of the moment conditions (B.1) using our sub-domain approximation (27) as,

$$\sum_j (\xi_i - x_j)^q \sum_{k=1}^{2} w_{ii_k} d(x_j - x_{i_k}^0) = \sum_{k=1}^{2} w_{ii_k} \sum_j (\xi_i - x_j)^q d(x_{i_k}^0 - x_j) \tag{B.2}$$

Now, we note that, interpolation using the linear two-point hat function described in Sec. 3.1.2 is equivalent to performing a linear interpolation with $n_p = 2$ nearest neighboring points in 1D. Therefore, the location of the body point ξ_i can be exactly expressed using the neighboring sub-domain points and their corresponding linear interpolation weights as,

$$\xi_j = \sum_{l=1}^{2} w_{ii_l} x_{i_l}^0 \tag{B.3}$$

The substitution of Eq. (B.3) into the right hand side of Eq. (B.2) yields,

$$\begin{aligned} \sum_j (\xi_i - x_j)^q \sum_{k=1}^{2} w_{ii_k} d(x_j - x_{i_k}^0) &= \sum_{k=1}^{2} w_{ii_k} \sum_j \left(\sum_{l=1}^{2} w_{ii_l} x_{i_l}^0 - x_j \right)^q d(x_{i_k}^0 - x_j) \\ &= \sum_{k=1}^{2} w_{ii_k} \sum_j \left(\sum_{l=\{k,k'\}} w_{ii_l} x_{i_l}^0 - x_j \right)^q d(x_{i_k}^0 - x_j) \end{aligned} \tag{B.4}$$

In the above, k' is defined as $k' = 2$ when $k = 1$ and vice versa. On further mathematical manipulation,

$$\begin{aligned} \sum_j (\xi_i - x_j)^q \sum_{k=1}^{2} w_{ii_k} d(x_j - x_{i_k}^0) &= \sum_{k=1}^{2} w_{ii_k} \sum_j \left(\sum_{l=\{k,k'\}} w_{ii_l} x_{i_l}^0 + w_{ii_{k'}} x_{i_k}^0 - w_{ii_{k'}} x_{i_k}^0 - x_j \right)^q d(x_{i_k}^0 - x_j) \\ &= \sum_{k=1}^{2} w_{ii_k} \sum_j \left(\sum_{l=\{k,k'\}} w_{ii_l} x_{i_k}^0 + w_{ii_{k'}} (x_{i_{k'}}^0 - x_{i_k}^0) - x_j \right)^q d(x_{i_k}^0 - x_j) \\ &= \sum_{k=1}^{2} w_{ii_k} \sum_j \left((x_{i_k}^0 - x_j) + (w_{ii_{k'}} (k' - k) \Delta x) \right)^q d(x_{i_k}^0 - x_j) \end{aligned} \tag{B.5}$$

Now, we check the individual moment conditions by substituting $q = \{0, 1, 2, 3\}$ into Eq. (B.5). When $q = 0$, Eq. (B.5) becomes,

$$\sum_j \sum_{k=1}^{2} w_{ii_k} d(x_j - x_{i_k}^0) = \sum_{k=1}^{2} w_{ii_k} \sum_j d(x_{i_k}^0 - x_j) = \sum_{k=1}^{2} w_{ii_k} = 1 \tag{B.6}$$

On going from the second to the third term above, we have used the fact that the underlying delta function $d(\cdot)$ satisfies the zeroth moment condition. Eq. (B.6) proves that the interpolated delta function satisfies the zeroth moment condition. Next, when $q = 1$, Eq. (B.5) becomes,

$$\begin{aligned}
\sum_j (\xi_i - x_j) \sum_{k=1}^{2} w_{ii_k} d(x_j - x_{i_k}^0) &= \sum_{k=1}^{2} w_{ii_k} \sum_j \left((x_{i_k}^0 - x_j) + (w_{ii_{k'}} (k' - k) \Delta x) \right) d(x_{i_k}^0 - x_j) \\
&= \sum_{k=1}^{2} w_{ii_k} \left(\sum_j (x_{i_k}^0 - x_j) d(x_{i_k}^0 - x_j) + (w_{ii_{k'}} (k' - k) \Delta x) \sum_j d(x_{i_k}^0 - x_j) \right) \\
&= \sum_{k=1}^{2} w_{ii_k} w_{ii_{k'}} (k' - k) \Delta x = 0
\end{aligned} \tag{B.7}$$

Again, on going from the second to the third line in the above equation, we have used the fact that $d(\cdot)$ satisfies the zeroth and first moment conditions. The last line simply involves expanding the summation by noting that $k' = 2$ when $k = 1$ and vice versa. Eq. (B.7) proves that the interpolated delta function satisfies the first moment condition. Next, when $q = 2$, Eq. (B.5) becomes,

$$\begin{aligned}
\sum_j (\xi_i - x_j)^2 \sum_{k=1}^{2} w_{ii_k} d(x_j - x_{i_k}^0) &= \sum_{k=1}^{2} w_{ii_k} \sum_j \left((x_{i_k}^0 - x_j) + (w_{ii_{k'}} (k' - k) \Delta x) \right)^2 d(x_{i_k}^0 - x_j) \\
&= \sum_{k=1}^{2} w_{ii_k} \left(\sum_j (x_{i_k}^0 - x_j)^2 d(x_{i_k}^0 - x_j) + \sum_j 2 (x_{i_k}^0 - x_j) (w_{ii_{k'}} (k' - k) \Delta x) d(x_{i_k}^0 - x_j) \right. \\
&\qquad \left. + \sum_j (w_{ii_{k'}} (k' - k) \Delta x)^2 d(x_{i_k}^0 - x_j) \right) \\
&= \sum_{k=1}^{2} w_{ii_k} \left(K + 0 + w_{ii_{k'}}^2 \Delta x^2 \right) = K + \Delta x^2 w_{ii_1} w_{ii_2}
\end{aligned} \tag{B.8}$$

Again, on going from the second to the third line in the above equation, we have used the fact that $d(\cdot)$ satisfies the zeroth, first and second moment conditions. Eq. (B.8) proves that the interpolated delta function satisfies the second moment condition with second order accuracy since $0 < w_{ii_1} w_{ii_2} < 1$, as the individual interpolation weights are bounded and finite, $0 < w_{ii_k} < 1$. Finally, when $q = 3$, Eq. (B.5) becomes,

$$\begin{aligned}
&\sum_j (\xi_i - x_j)^3 \sum_{k=1}^{2} w_{ii_k} d(x_j - x_{i_k}^0) \\
&= \sum_{k=1}^{2} w_{ii_k} \sum_j \left((x_{i_k}^0 - x_j) + (w_{ii_{k'}} (k' - k) \Delta x) \right)^3 d(x_{i_k}^0 - x_j) \\
&= \sum_{k=1}^{2} w_{ii_k} \left(\sum_j (x_{i_k}^0 - x_j)^3 d(x_{i_k}^0 - x_j) + \sum_j 3 (x_{i_k}^0 - x_j)^2 (w_{ii_{k'}} (k' - k) \Delta x) d(x_{i_k}^0 - x_j) \right. \\
&\qquad \left. + \sum_j 3 (x_{i_k}^0 - x_j) (w_{ii_{k'}} (k' - k) \Delta x)^2 d(x_{i_k}^0 - x_j) + \sum_j (w_{ii_{k'}} (k' - k) \Delta x)^3 d(x_{i_k}^0 - x_j) \right) \\
&= \sum_{k=1}^{2} w_{ii_k} \left(0 + 3K (w_{ii_{k'}} (k' - k) \Delta x) + 0 + (w_{ii_{k'}} (k' - k) \Delta x)^3 \right) = \Delta x^3 w_{ii_1} w_{ii_2} (w_{ii_1} - w_{ii_2})
\end{aligned} \tag{B.9}$$

Again, on going from the second to the third line in the above equation, we have used the fact that $d(\cdot)$ satisfies the zeroth to third moment conditions and $-1 < w_{ii_1} w_{ii_2} (w_{ii_1} - w_{ii_2}) < 1$. Eq. (B.9) proves that the interpolated delta function satisfies the third moment condition with third order accuracy.

Eq. (B.6)–(B.9) prove that the interpolated delta function satisfies zeroth to third moment conditions with at least second order accuracy provided that the underlying delta function being interpolated satisfies these conditions. Higher order moment conditions can be proved similarly. This minimum second order accuracy is in accordance with the maximum second order accuracy of the immersed boundary method.

Appendix C. Convergence of the sub-domain based IB approach

In this section, we analyze the convergence properties of the two approximations involved in our proposed approach: (a) sub-domain approximation, $E_n^{(k)} \approx P_n^{(k)} E_0$ and (b) approximating dense B as sparse B'.

Firstly, we consider the sub-domain approximation, $E_n^{(k)} \approx P_n^{(k)} E_0$, where the delta functions in $E_n^{(k)}$ are obtained by interpolating the delta functions corresponding to the sub-domain points in E_0. This interpolation is second order accurate in the sub-domain grid spacing, $\mathcal{O}((\Delta x^0)^2)$ as mentioned in Sec. 3.1.2. Furthermore, the sub-domain grid spacing is set to be equal to the flow domain spacing, $\Delta x^0 = \Delta x$ (*cf.* Sec. 3.1.1). Therefore, the sub-domain approximation converges to true $E_n^{(k)}$ as $\mathcal{O}(\Delta x^2)$.

Next, we analyze the convergence of the sparsity approximation which is built upon the sub-domain approximation. The full matrix B can be exactly expressed as $B = B' + \Xi$, where Ξ is the matrix containing the eliminated elements of B after passing through the drop tolerance filter. The surface stress equation involving the full B (*i.e.* Eq. (23) substituted into Eq. (17)) can be expressed in compact form as, $PBP^T f = r$, where r is a generic right hand side vector, the n and k indices are dropped for neatness, and the structural operators are omitted for simplicity. The residual of this equation can be written as,

$$R(f) = PBP^T - r = PB'P^T f - r + P\Xi P^T f = R'(f) + P\Xi P^T f \tag{C.1}$$

In the above, $R(f)$ denotes the residual involving the sub-domain approximation only, while $R'(f)$ denotes the residual that incorporates the sparsity approximation as well. In our proposed approach which employs both the approximations, we are solving for the latter residual, $R'(f) = 0$. Therefore, $R(f)$ can be simplified as,

$$R(f) = P\Xi P^T f \tag{C.2}$$

Recall that the elements of Ξ are determined by using the drop tolerance parameter, ϵ. In other words, as $\epsilon \to 0$, $\Xi \to 0$. Therefore, the residual $R(f)$ associated with the purely sub-domain approximation converges to zero as $\epsilon \to 0$.

To summarize, the sub-domain approximation converges to true $E_n^{(k)}$ as $\mathcal{O}(\Delta x^2)$, while the added sparsity approximation is consistent in the sense that the residual of the surface stress equation with the sub-domain approximation converges to zero as $\epsilon \to 0$.

Appendix D. Proposed sub-domain based IB approach in primitive variables

In this section, we derive the sub-domain based IB approach that treats the fluid in primitive variables. Following the appendix in Goza and Colonius [15], the governing equations in the primitive variables (1)–(7) are fully discretized in space and time, subjected to a block-LU decomposition and an FSI iterative scheme are applied. The resulting system of equations is

$$u^* = A^{-1}\tilde{r}_n^f - A^{-1}G(G^TA^{-1}G)^{-1}(G^TA^{-1}\tilde{r}_n^f - \tilde{r}_n^p) \tag{D.1}$$

$$p^* = (G^TA^{-1}G)^{-1}(G^TA^{-1}\tilde{r}_n^f - \tilde{r}_n^p) \tag{D.2}$$

$$\begin{aligned}\left(E_{n+1}^{(k)}A^{-1}(I - G(G^TA^{-1}G)^{-1}G^TA^{-1})E_{n+1}^{(k)T} + \frac{2\Delta s}{\Delta t}S_{t,n+1}^{(k)T}J_t^{-1}S_{t,n+1}^{(k)} + \frac{2}{\Delta t}\hat{I}_d^T J_d^{(k)^{-1}} S_{d,n+1}^{(k)}\right) f_{n+1}^{(k+1)} = \\ E_{n+1}^{(k)}U^* - r^{c(k)} + S_{t,n+1}^{(k)T}\left(r^{\theta(k)} - \frac{2}{\Delta t}J_t^{-1}r^{\phi(k)}\right) + \hat{I}_d^T\left(r^{\chi(k)} - \frac{2}{\Delta t}J_d^{(k)^{-1}}r^{\zeta(k)}\right)\end{aligned} \tag{D.3}$$

$$\Delta\theta = J_t^{-1}\left(r^{\phi(k)} + \Delta s S_{t,n+1}^{(k)} f_{n+1}^{(k+1)}\right) \tag{D.4}$$

$$\Delta\chi_d = J_d^{(k)^{-1}}\left(r^{\zeta(k)} + S_{d,n+1}^{(k)} f_{n+1}^{(k+1)}\right) \tag{D.5}$$

$$p_{n+1} = p^* - (G^TAG)^{-1}G^TA^{-1}E_{n+1}^{(k)}f_{n+1} \tag{D.6}$$

$$u_{n+1} = u^* - A^{-1}(I - G(G^TA^{-1}G)^{-1}G^TA^{-1})E_{n+1}^{(k)}f_{n+1} \tag{D.7}$$

Here, G denotes the gradient operator and the right hand side terms $\tilde{r}_n^f$ and $\tilde{r}_n^p$ are analogous to those in Appendix A. The above equations are written in a similar fractional step format of the vorticity-streamfunction formulation (16)–(20). Furthermore, the FSI iterations identified by the superscript (k) are applied only on the small dimensional equations (D.3)–(D.5), which are solved for the surface stress and body position.

Even though the surface stress equation (D.3) is small-dimensional, it contains an embedded large dimensional solve in the form of $A^{-1}(I - G(G^T A^{-1} G)^{-1} G^T A^{-1})$ resulting in a severe computational bottleneck similar to that discussed in Sec. 2.3. This bottleneck is mitigated by using the sub-domain approximation, $E_{n+1}^{(k)} \approx P_{n+1}^{(k)} E_0$, where all the terms and sub-domain involved are exactly the same as that defined in Sec. 3. Now, the expensive solution procedure of solving Eq. (D.3) can be rewritten as,

$$E_{n+1}^{(k)} A^{-1}(I - G(G^T A^{-1} G)^{-1} G^T A^{-1}) E_{n+1}^{(k)T} \approx P_{n+1}^{(k)} \left(E_0 A^{-1}(I - G(G^T A^{-1} G)^{-1} G^T A^{-1}) E_0^T \right) P_{n+1}^{(k)T} = P_{n+1}^{(k)} \tilde{B} P_{n+1}^{(k)T} \tag{D.8}$$

where $\tilde{B} = E_0 A^{-1}(I - G(G^T A^{-1} G)^{-1} G^T A^{-1}) E_0^T$ is time-invariant and generally small-dimensional, thereby allowing us to precompute and store $\tilde{B}$ once and for all. Additionally, since $P_{n+1}^{(k)}$ is sparse, evaluation of $P_{n+1}^{(k)} \tilde{B} P_{n+1}^{(k)T}$ is performed at minimal computational cost that scales only with the small number of body interface points.

Similar to B, $\tilde{B}$ can be sparsified using the drop tolerance filtering algorithm in Sec. 3.2.2. However, the value of drop tolerance to be used for sparsifying $\tilde{B}$ could be different from the suggested $\epsilon = 0.007$ for the vorticity-streamfunction formulation and therefore, would have to be re-calibrated to strike the balance between accuracy of solutions and storage requirements.

References

[1] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[2] D. Kim, H. Choi, Immersed boundary method for flow around an arbitrarily moving body, J. Comput. Phys. 212 (2006) 662–680.
[3] K. Taira, T. Colonius, The immersed boundary method: a projection approach, J. Comput. Phys. 225 (2007) 2118–2137.
[4] W. Kim, H. Choi, Immersed boundary methods for fluid-structure interaction: a review, Int. J. Heat Fluid Flow 75 (2019) 301–309.
[5] W.-X. Huang, F.-B. Tian, Recent trends and progress in the immersed boundary method, Proc. Inst. Mech. Eng., Part C, J. Mech. Eng. Sci. 233 (2019) 7617–7636.
[6] W. Kim, I. Lee, H. Choi, A weak-coupling immersed boundary method for fluid–structure interaction with low density ratio of solid to fluid, J. Comput. Phys. 359 (2018) 296–311.
[7] L. Wang, F.-B. Tian, J.C. Lai, An immersed boundary method for fluid–structure–acoustics interactions involving large deformations and complex geometries, J. Fluids Struct. 95 (2020) 102993.
[8] P. Causin, J.-F. Gerbeau, F. Nobile, Added-mass effect in the design of partitioned algorithms for fluid–structure problems, Comput. Methods Appl. Mech. Eng. 194 (2005) 4506–4527.
[9] C. Förster, W.A. Wall, E. Ramm, Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows, Comput. Methods Appl. Mech. Eng. 196 (2007) 1278–1293.
[10] I. Borazjani, L. Ge, F. Sotiropoulos, Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies, J. Comput. Phys. 227 (2008) 7587–7620.
[11] F.-B. Tian, H. Dai, H. Luo, J.F. Doyle, B. Rousseau, Fluid–structure interaction involving large deformations: 3D simulations and applications to biological systems, J. Comput. Phys. 258 (2014) 451–469.
[12] M.D. de Tullio, G. Pascazio, A moving-least-squares immersed boundary method for simulating the fluid–structure interaction of elastic bodies with arbitrary thickness, J. Comput. Phys. 325 (2016) 201–225.
[13] J. Degroote, K.-J. Bathe, J. Vierendeels, Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction, Comput. Struct. 87 (2009) 793–801.
[14] C. Wang, J.D. Eldredge, Strongly coupled dynamics of fluids and rigid-body systems with the immersed boundary projection method, J. Comput. Phys. 295 (2015) 87–113.
[15] A. Goza, T. Colonius, A strongly-coupled immersed-boundary formulation for thin elastic structures, J. Comput. Phys. 336 (2017) 401–411.
[16] U. Lācis, K. Taira, S. Bagheri, A stable fluid–structure-interaction solver for low-density rigid bodies using the immersed boundary projection method, J. Comput. Phys. 305 (2016) 300–318.
[17] S. Tschisgale, J. Fröhlich, An immersed boundary method for the fluid-structure interaction of slender flexible structures in viscous fluid, J. Comput. Phys. 423 (2020) 109801.
[18] J. Yang, F. Stern, A non-iterative direct forcing immersed boundary method for strongly-coupled fluid-solid interactions, J. Comput. Phys. 295 (2015) 779–804.
[19] L. Xu, F.-B. Tian, J. Young, J.C. Lai, A novel geometry-adaptive Cartesian grid based immersed boundary–lattice Boltzmann method for fluid–structure interactions at moderate and high Reynolds numbers, J. Comput. Phys. 375 (2018) 22–56.
[20] T. Colonius, K. Taira, A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions, Comput. Methods Appl. Mech. Eng. 197 (2008) 2131–2146.
[21] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[22] A. Goza, S. Liska, B. Morley, T. Colonius, Accurate computation of surface stresses and forces with immersed boundary methods, J. Comput. Phys. 321 (2016) 860–873.
[23] X. Yang, X. Zhang, Z. Li, G.-W. He, A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations, J. Comput. Phys. 228 (2009) 7821–7836.
[24] N. Jovanovic, D. Keyes, K.G. Prasad, J. Kane, Drop tolerance ILU preconditioned for iterative solution techniques in boundary element analysis, WIT Trans. Model. Simul. 1 (1970).
[25] G. Strang, The discrete cosine transform, SIAM Rev. 41 (1999) 135–147.
[26] S. Wang, X. Zhang, An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows, J. Comput. Phys. 230 (2011) 3479–3499.

[27] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, D. Karpeyev, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, R.T. Mills, T. Munson, K. Rupp, P. Sanan, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Users Manual, Technical Report ANL-95/11 - Revision 3.14, Argonne National Laboratory, 2020, https://www.mcs.anl.gov/petsc.
[28] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, Proc. IEEE 93 (2005) 216–231, Special issue on "Program Generation, Optimization, and Platform Adaptation".
[29] J. Toomey, J.D. Eldredge, Numerical and experimental study of the fluid dynamics of a flapping wing with low order flexibility, Phys. Fluids 20 (2008) 073603.
[30] M.E. Rosti, M. Omidyeganeh, A. Pinelli, Passive control of the flow around unsteady aerofoils using a self-activated deployable flap, J. Turbul. 19 (2018) 204–228.
[31] C. Duan, J. Waite, A. Wissa, Design optimization of a covert feather-inspired deployable structure for increased lift, in: Applied Aerodynamics Conference, AIAA Aviation Forum, 2018, p. 3174.
[32] J.D. Eldredge, A method of immersed layers on Cartesian grids, with application to incompressible flows, J. Comput. Phys. 448 (2022) 110716.
[33] R. Meyer, W. Hage, D.W. Bechert, M. Schatz, T. Knacke, F. Thiele, Separation control by self-activated movable flaps, AIAA J. 45 (2007) 191–199.
[34] C. Zhu, J.H. Seo, V. Vedula, R. Mittal, A highly scalable sharp-interface immersed boundary method for large-scale parallel computers, in: 23rd AIAA Computational Fluid Dynamics Conference, 2017, p. 3622.
[35] S. Wang, G. He, X. Zhang, Parallel computing strategy for a flow solver based on immersed boundary method and discrete stream-function formulation, Comput. Fluids 88 (2013) 210–224.
[36] J.W. Banks, W.D. Henshaw, D.W. Schwendeman, Q. Tang, A stable partitioned FSI algorithm for rigid bodies and incompressible flow in three dimensions, J. Comput. Phys. 373 (2018) 455–492.
[37] B. Yildirim, S. Lin, S. Mathur, J.Y. Murthy, A parallel implementation of fluid–solid interaction solver using an immersed boundary method, Comput. Fluids 86 (2013) 251–274.
[38] J.K. Wiens, J.M. Stockie, An efficient parallel immersed boundary algorithm using a pseudo-compressible fluid solver, J. Comput. Phys. 281 (2015) 917–941.
[39] Z. Wang, J. Fan, K. Luo, Parallel computing strategy for the simulation of particulate flows with immersed boundary method, Sci. China, Ser. E, Technol. Sci. 51 (2008) 1169–1176.
[40] M. Frigo, S.G. Johnson, Parallel FFTW on a Sun HPC 5000, http://www.fftw.org/parallel/xolas.html. (Accessed 30 September 2010).
[41] S. Liska, T. Colonius, A fast immersed boundary method for external incompressible viscous flows using lattice Green's functions, J. Comput. Phys. 331 (2017) 257–279.