Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 3114–3140
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A fixed-mesh method for incompressible flow–structure systems with finite solid deformations

Hong Zhao [a], Jonathan B. Freund [a,b,*], Robert D. Moser [c]

[a] Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, 1206 West Green Street, Urbana, IL 61801, United States
[b] Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, 306 Talbot Laboratory, 104 South Wright Street, Urbana, IL 61801, United States
[c] Department of Mechanical Engineering, Institute for Computational Engineering and Science, University of Texas at Austin, 1 University Station C2200, Austin, TX 78712, United States

Received 2 April 2007; received in revised form 5 November 2007; accepted 13 November 2007
Available online 23 November 2007

Abstract

A fixed-mesh algorithm is proposed for simulating flow–structure interactions such as those occurring in biological systems, in which both the fluid and solid are incompressible and the solid deformations are large. Several of the well-known difficulties in simulating such flow–structure interactions are avoided by formulating a single set of equations of motion on a fixed Eulerian mesh. The solid's deformation is tracked to compute elastic stresses by an overlapping Lagrangian mesh. In this way, the flow–structure interaction is formulated as a distributed body force and singular surface force acting on an otherwise purely fluid system. These forces, which depend on the solid elastic stress distribution, are computed on the Lagrangian mesh by a standard finite-element method and then transferred to the fixed Eulerian mesh, where the joint momentum and continuity equations are solved by a finite-difference method. The constitutive model for the solid can be quite general. For the force transfer, standard immersed-boundary and immersed-interface methods can be used and are demonstrated. We have also developed and demonstrated a new projection method that unifies the transfer of the surface and body forces in a way that exactly conserves momentum; the interface is still effectively sharp for this approach. The spatial convergence of the method is observed to be between first- and second-order, as in most immersed-boundary methods for membrane flows. The algorithm is demonstrated by the simulations of an advected elastic disk, a flexible leaflet in an oscillating flow, and a model of a swimming jellyfish.

1. Introduction

We propose a new algorithm to simulate the type of flow–structure interactions that often occur in biological systems, such as in cardiovascular flow or swimming animals. Here, the deformations of the solid material

* Corresponding author. Address: Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, 1206 West Green Street, Urbana, IL 61801, United States. Tel.: +1 217 244 7729.
E-mail addresses: hongzhao@uiuc.edu (H. Zhao), jbfreund@uiuc.edu (J.B. Freund), rmoser@mail.utexas.edu (R.D. Moser).

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.11.019

are in general large and tightly coupled to the flowing fluid. The solid and fluid components are also typically incompressible. This type of system is particularly challenging to simulate because the governing equations are different in the two regions and the interface location must be solved for simultaneously with the dynamics in both regions in a way that enforces both kinematic (no slip) and dynamic (stress matched) boundary conditions at the interface. Incompressibility must also be enforced in both regions, which is a fundamentally different problem in each.

We have designed an algorithm that is accurate and efficient, and has provable properties despite the difficulties associated with these two distinct regions by formulating a unified equation of motion, in a straightforward way, that is nearly the same in the two regions. It is formulated in an Eulerian frame, as is more common for simulating fluids, and it is discretized on a Cartesian mesh. The difference between the two regions is in the stress formulation, which depends upon the reference configuration in the elastic solid. This configuration is tracked with a separate Lagrangian mesh, which effectively overlays the Cartesian mesh. In this formulation, the flow–structure interaction reduces to a singular surface force **F** and a body force **B**, which are first computed on the Lagrangian mesh, and then transferred to the momentum equation solved on the Eulerian mesh. For the momentum equation solver, we used a finite-difference fractional step algorithm [26] that has been widely used for incompressible Newtonian fluid flows. Cartesian meshes are especially attractive in this approach due to their simplicity, which leads to fast computation techniques such as approximate factorization methods and fast Fourier transforms. The solid stresses are evaluated with a finite-element method that imposes little restriction on the constitutive model for the solid. We demonstrate the method for neo-Hookean solids.

Body-fitted moving meshes are another means of representing the solid and fluid regions separately, and the matching between the two regions is straightforward if it occurs along mesh lines. However, for the large deformations we seek to simulate, these meshes often need to be locally or globally regenerated to maintain good mesh quality, which is usually computationally expensive and not always robust. Any interpolation from the old mesh to the new one incurs extra computational cost and error [20,24]. Also, in standard body-fitted mesh formations, the implementation of fluid–solid coupling is loose, meaning that there is no guarantee that at the end of any particular time step both solvers predict precisely the same location of the interface. Standard iteration schemes to resolve this are equivalent to a block Gauss–Seidel iteration, and so tend to be inefficient and are not guaranteed to converge [15]. The fluid and solid motions can also be jointly solved, resulting in better coupling. For example, in the context of the finite-element method, a joint velocity space can be defined such that the velocity is continuous across the interface; a weak formulation can then be derived from a virtual-work principle, thereby guaranteeing continuity of surface traction. This coupling algorithm was used by Hu and coworkers in simulating particulate flows [19,20]. Such coupling algorithms are in general more difficult to design for non-body-fitted mesh methods because of the mismatch between the mesh and the interface. One example is the distributed Lagrange multiplier (DLM) method proposed by Glowinski et al. [11], which constrains the fluid and solid velocities to match. The DLM method was originally designed for simulating particulate flows in which the solid particles are rigid [11,12], and it has been generalized to treat flow interacting with a flexible body [54]. Other notable flow–structure interaction algorithms of the finite-element type include the immersed finite-element method (IFEM) [33,52] and the material-point method [46,47], which is discussed in more detail in Section 3.3.3.

Our formulation is similar in spirit to a class of finite-volume or finite-difference fixed-mesh algorithms for simulation of fluid flow in complex geometries on fixed Cartesian meshes [4,7,48,49] but takes the important step of including dynamically deforming solids. In this sense, it is more similar to Peskin's immersed-boundary method (IBM) [3,9,40,41,46] and LeVeque and Li's immersed-interface method (IIM) [30,31]. In those methods, the fluid velocity and pressure are defined on a fixed Cartesian mesh, and the membrane is represented by a set of Lagrangian points convected by the fluid velocity field. The membrane affects the surrounding fluid flow field via a singular surface force, whose density is determined by the force balance on the membrane surface. The governing equations of motion for the fluid on both sides of the membrane can thus be formulated as a single Navier–Stokes equation supplemented by a surface force term. For our formulation, we show that the singular component **F** of the solid's stress on the fluid can be handled as in either of these methods, but we also develop an accurate projection-based unified means of transferring both this singular component **F** and the body force **B** to the Cartesian mesh in a way that exactly guarantees conservation of momentum. In our

formulation, the interface is not smeared beyond the neighboring mesh points as it is for the IBM, and at the same time it avoids the challenge of taking derivatives of the local interface shape as needed for the IIM approach. It does not match the formal order of accuracy that can be achieved with the IIM approach (with considerable difficulty), but it has guaranteed conservation of momentum, maintains a sharp interface, and is easier to implement, especially in three dimensions where the topological representation of the surface becomes challenging for the IIM.

In the following, Section 2 sets up the details of the approach and the analytical formulation, and Section 3 provides the details of its numerical solution. The accuracy and convergence are assessed in Section 4, which also provides more complex demonstrations of the method. Extensions and generalization are discussed along with the summary of the method in Section 5.

2. Problem definition and methodology

We consider flow–structure interaction systems in which both the fluid and the solid are incompressible and have the same density. This assumption is a good approximation for many biological systems in which the solids are tissues with densities close to that of the surrounding fluid (usually mostly water), and are soft but highly resistant to any local volume changes. For most of the analysis, we also assume that the solid and fluid have the same viscosity—an assumption that simplifies the treatment of the traction boundary condition at the interface. With this assumption, and in the flows of interest, the velocity gradient in the solid will be smaller than that in the fluid, so the added viscous forces in the solid are negligible compared to the solid elastic forces. Additional viscosity can be added to the solid in a straightforward way, which we discuss in Section 5.

The geometric notation defining the flow–structure system is shown in Fig. 1. The computational domain Ω is divided into distinct fluid Ω_f and solid Ω_s regions:

$$\Omega_\mathrm{f} = \Omega_\mathrm{f}^1 \cup \Omega_\mathrm{f}^2 \cup \cdots \quad \text{and} \quad \Omega_\mathrm{s} = \Omega_\mathrm{s}^1 \cup \Omega_\mathrm{s}^2 \cup \cdots, \tag{1}$$

such that

$$\Omega_\mathrm{s} \cup \Omega_\mathrm{f} = \Omega, \qquad \Omega_\mathrm{s} \cap \Omega_\mathrm{f} = \emptyset. \tag{2}$$

The solid and fluid are separated by an interface $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \cdots$, whose normal $\mathbf{n}$ points from the solid into the fluid.

The motion of both the fluid and the solid is governed by mass and momentum conservation equations, which we choose to write in a generic Eulerian form,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nabla \cdot \boldsymbol{\tau}, \tag{3a}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{3b}$$

where $\mathbf{u}$ is the velocity, p is the pressure, and $\boldsymbol{\tau}$ is the deviatoric stress tensor. In addition, the solid moves at local velocity

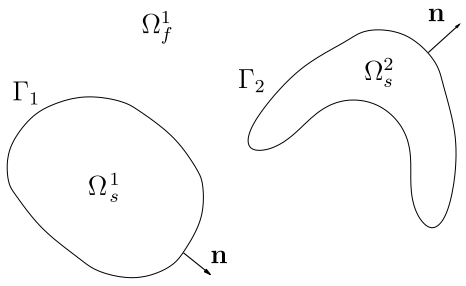


Fig. 1. Notation for the system.

$$\frac{\partial \mathbf{x}(\mathbf{X},t)}{\partial t} = \mathbf{u}(\mathbf{x},t), \tag{4}$$

where $\mathbf{x}$ is the current coordinate of a solid material point and $\mathbf{X}$ is its coordinate at a reference state, which we take to be stress-free. All physical quantities are non-dimensionalized by density and appropriate length and velocity scales. The fluid and solid motions are coupled at their interface by the no-slip condition and the balance of surface traction:

$$[\![\mathbf{u}]\!] = 0, \tag{5a}$$

$$[\![-p\mathbf{n} + \boldsymbol{\tau}\cdot\mathbf{n}]\!] = 0, \tag{5b}$$

where the notation $[\![q]\!]$ denotes a jump from solid to fluid across the interface,

$$[\![q]\!] = q|_{\text{fluid}} - q|_{\text{solid}}. \tag{6}$$

The fluid is assumed to be Newtonian with viscous stress

$$\boldsymbol{\tau}_{\text{f}} = \mu_{\text{f}}(\nabla\mathbf{u} + \nabla\mathbf{u}^T), \tag{7}$$

where μ_{f} is the non-dimensional dynamic viscosity of fluid. The solid deviatoric stress has both elastic and viscous components:

$$\boldsymbol{\tau}_{\text{s}} = \boldsymbol{\tau}_{\text{elas}} + \boldsymbol{\tau}_{\text{visc}}, \tag{8}$$

where the viscous part $\boldsymbol{\tau}_{\text{visc}}$ by our assumption has the same form as the fluid viscous stress (7). A neo-Hookean elasticity model is used for the elastic stress,

$$\boldsymbol{\tau}_{\text{elas}} = \mu_{\text{s}}(\mathbf{A}\cdot\mathbf{A}^T - \mathbf{I}), \tag{9}$$

where μ_{s} is the elastic constant and

$$\mathbf{A} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \tag{10}$$

is the deformation gradient tensor, but we shall see that our formulation for flow–structure coupling is independent of the solid's constitutive properties.

For our assumed incompressible Newtonian fluid, the momentum equation is, of course, the standard Navier–Stokes equation (N–S),

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = -\nabla p + \mu_{\text{f}}\nabla^2\mathbf{u}. \tag{11}$$

In the solid, it is identical aside from an extra term $\nabla\cdot\boldsymbol{\tau}_{\text{elas}}$,

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = -\nabla p + \mu_{\text{f}}\nabla^2\mathbf{u} + \nabla\cdot\boldsymbol{\tau}_{\text{elas}}. \tag{12}$$

From the perspective of the momentum Eqs. (11) and (12), the term $\nabla\cdot\boldsymbol{\tau}_{\text{elas}}$ can be regarded as a body force with support only in Ω_{s}. The elastic stress term in the jump condition (5b) can thus be moved to the right-hand side to yield

$$[\![-p\mathbf{n} + \mu_{\text{f}}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)\cdot\mathbf{n}]\!] = \boldsymbol{\tau}_{\text{elas}}\cdot\mathbf{n}. \tag{13}$$

The right-hand side $\boldsymbol{\tau}_{\text{elas}}\cdot\mathbf{n}$ can be regarded as a surface force density acting on the surrounding fluid, which is similar to that exerted by an elastic membrane. These observations suggest a combined momentum equation for the entire system,

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = -\nabla p + \mu_{\text{f}}\nabla^2\mathbf{u} + \mathbf{B} + \mathbf{F}, \tag{14}$$

where $\mathbf{B}$ is the body force of the form $\nabla\cdot\boldsymbol{\tau}_{\text{elas}}$ as in (12) and $\mathbf{F}$ is a surface force arising from (13). The body force $\mathbf{B}$ has support in Ω_{s},

$$\mathbf{B}(\mathbf{x},t) = \chi_{\text{s}}(\mathbf{x},t)\,\nabla\cdot\boldsymbol{\tau}_{\text{elas}}, \tag{15}$$

where χ_s is the characteristic function of Ω_s (i.e., it is 1 in Ω_s and 0 elsewhere). The surface force $\mathbf{F}$ arising from (13) can be written as

$$\mathbf{F}(\mathbf{x},t) = \int_\Gamma \hat{\mathbf{f}}(\xi,t)\delta(\mathbf{x}-\mathbf{r}(\xi,t))\,\mathrm{d}a(\xi), \tag{16}$$

where ξ is a coordinate parameterizing the interface Γ, $\mathbf{r}(\xi,t)$ is the interface position, and the surface force density $\hat{\mathbf{f}}$ is

$$\hat{\mathbf{f}} = -\boldsymbol{\tau}_{\text{elas}} \cdot \mathbf{n}. \tag{17}$$

The equivalence of the combined momentum Eq. (14) and the original momentum Eq. (3a) with jump condition (13) can be easily proved by integration by parts.

3. Numerical algorithm

The complete numerical scheme for solving the system described by (14), (3b) and (4) includes four main components:

(1) A momentum equation solver for velocity and pressure on an Eulerian mesh.
(2) A means to compute forces **B** and **F** on the Lagrangian mesh.
(3) A means to transfer **B** and **F** to the Eulerian mesh.
(4) Movement of the Lagrangian mesh consistent with the solid's incompressibility.

The following four sections discuss the methods and algorithms used and developed for each of these components. The formulation is general, but is presented here in two space dimensions for simplicity. Section 3.5 ties them all into a unified time advancement scheme.

3.1. Eulerian-mesh momentum equation solver

For simplicity, here the computational domain Ω is a two-dimensional rectangle, and is discretized by a uniform Cartesian mesh with mesh spacings h_x and h_y. The discrete velocity and pressure are staggered, as shown in Fig. 2. In this formulation, the discrete divergence and gradient operators are exactly adjoint, making the scheme inviscidly stable and non-dissipative for Navier–Stokes solutions [16,51]. The Cartesian mesh facilitates the use of fast Fourier transforms to solve the discrete pressure Poisson equation. Mesh uniformity is not a necessity: for example, local mesh refinement [34,44] could be included. When the domain boundary $\partial\Omega$ is irregular, a curvilinear mesh can provide greater geometric flexibility, but a Cartesian mesh can also still be used with Lagrange multipliers to enforce the boundary conditions on $\partial\Omega$, which is embedded within the mesh [25,48].

In the absence of solid, the governing equations of motion are the familiar incompressible Navier–Stokes equations. The convection term, the viscous term, and the pressure gradient term are all discretized by standard central finite-differences on the staggered mesh as proposed by Kim and Moin [26] and summarized in detail by Zhao [55]. The forces **B** and **F**, after being transferred to the Eulerian mesh, are discretized pointwise at velocity mesh points, so they are straightforwardly added to the right-hand side of the discrete momentum

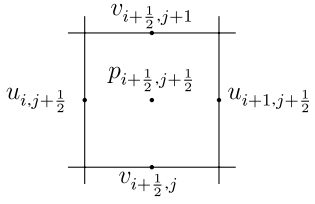


Fig. 2. Variable definition on the staggered Cartesian mesh.

equation. The discretized equations are integrated in time by a three-substep scheme [45]. Every substep, a projection is employed to enforce the divergence-free constraint, as discussed in Section 3.5.

3.2. Computation of solid elastic stress

The solid deformation is tracked by a Lagrangian mesh, upon which the solid elastic stress is computed via standard isoparametric finite-element interpolations. The Lagrangian mesh makes it easy to simulate periodic and reversible solid deformations commonly occurring in biomechanics; it is, of course, less capable of handling large plastic deformations and topological changes such as fractures, for which a purely Eulerian approach [50] is more appropriate. The Lagrangian mesh points $\mathbf{x}$ that track the solid are each referenced to fixed points $\mathbf{X}$, and the deformation tensor $\mathbf{A}$ within each element e is computed as

$$\mathbf{A} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \right)^{-1} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{pmatrix} \begin{pmatrix} \frac{\partial X}{\partial \xi} & \frac{\partial X}{\partial \eta} \\ \frac{\partial Y}{\partial \xi} & \frac{\partial Y}{\partial \eta} \end{pmatrix}^{-1}, \tag{18}$$

where $\boldsymbol{\xi} = (\xi, \eta)$ is the coordinate in a standard reference element. The derivatives with respect to (ξ, η) are computed via isoparametric interpolation,

$$\frac{\partial \mathbf{x}}{\partial \xi} = \sum_j \mathbf{x}_j^e \frac{\partial N_j^e(\xi, \eta)}{\partial \xi}, \quad \frac{\partial \mathbf{x}}{\partial \eta} = \sum_j \mathbf{x}_j^e \frac{\partial N_j^e(\xi, \eta)}{\partial \eta}, \tag{19}$$

where $\mathbf{x}_j^e$ is the coordinate of the jth vertex of element e, and $N_j^e(\xi, \eta)$ is the associated nodal shape function. For incompressible solids, simple linear triangular (T3) or bilinear quadrilateral (Q4) elements are known to lead to ill-conditioned stiffness matrices unless stabilized [21]. To avoid these so-called "locking" problems, we use six-node quadratic triangular (T6) elements or nine-node quadratic quadrilateral (Q9) elements.

Since isoparametric interpolants are only C^0-continuous on the global Lagrangian mesh, the computed $\mathbf{A}$ and thus $\boldsymbol{\tau}_{\text{elas}}$ are discontinuous between neighboring elements. This in turn leads to force singularities in $\nabla \cdot \boldsymbol{\tau}_{\text{elas}}$, which cannot be directly transferred to the Cartesian mesh. Doing so would damage numerical accuracy, reducing its formal order, and potentially cause numerical instability. Instead, the Zienkiewicz–Zhu [57] (ZZ) patch recovery method, which is a standard technique for computing stress field in finite-element methods, is used to rebuild a continuous $\boldsymbol{\tau}_{\text{elas}}$ field, as an approximation to the original elementwise discontinuous $\boldsymbol{\tau}_{\text{elas}}$. The density of the surface force $\mathbf{F}$ on Γ in (16) is then computed from the reconstructed $\boldsymbol{\tau}_{\text{elas}}$ and $\mathbf{n}$. To compute $\mathbf{B}$ in (15), the stress divergence $\nabla \cdot \boldsymbol{\tau}_{\text{elas}}$ is first computed in each element from the continuous $\boldsymbol{\tau}_{\text{elas}}$ field, and the ZZ patch method is applied again to recover a continuous $\nabla \cdot \boldsymbol{\tau}_{\text{elas}}$ in Ω_s.

3.3. Transfer of the elastic stress

Though our unified formulation eases the task of coupling the fluid and solid, the transfer of $\mathbf{B}$ and $\mathbf{F}$ must still be done consistently and accurately. There are two basic perspectives for undertaking this: we can attempt to minimize the difference between the numerical force distributions on the two meshes, or we can attempt to minimize local truncation errors in the final discretized momentum Eq. (14). Different force-distribution algorithms can be designed from these two perspectives, and their properties such as accuracy and momentum conservation will vary, with the preferred approach probably depending upon the objectives of any particular application. The ease of implementation, especially in three dimensions, is also an important issue. Three force transfer methods are proposed in the following subsections: an immersed-boundary method as proposed for membrane flows [40,41], an immersed-interface method, also as proposed for membrane flows [30,31], and a new projection method. All three are successful, and their relative merits are discussed in Section 3.3.4.

3.3.1. Immersed-boundary method

In application of this approach, the surface force $\mathbf{F}$ is approximated by a smooth body-force distribution with a narrow band of support around Γ [40,41]. Just as for membrane flows, we approximate the singular $\mathbf{F}$ by its convolution with a smooth kernel function d_h, for which we use a commonly selected form [28]:

$$d_h(\mathbf{x}) = \delta_{h_x}(x)\delta_{h_y}(y) \quad \text{with } \delta_h(r) = \begin{cases} \frac{1}{4h}\left(1+\cos\frac{\pi r}{2h}\right) & |r| \leqslant 2h, \\ 0 & |r| \geqslant 2h. \end{cases} \tag{20}$$

The mollified force is then

$$\mathbf{F}_h(\mathbf{x},t) = \int_\Gamma \hat{\mathbf{f}}(s,t) d_h(\mathbf{x} - \mathbf{r}(s,t))\,\mathrm{d}s, \tag{21}$$

where s is the arc length along the one-dimensional interface and $\hat{\mathbf{f}}$ is defined in (17). $\mathbf{F}_h$ is smooth and has support in four Cartesian mesh cells around the interface. Its pointwise values are well defined and can be straightforwardly computed at every velocity point on the Cartesian mesh.

The body force $\mathbf{B}$ is not part of the standard immersed-boundary formulations, but as computed by ZZ patch recovery, it is continuous so its interpolation to the Cartesian mesh is straightforward. For a Cartesian mesh point $\mathbf{x}_{\alpha,\beta} = (x_\alpha, y_\beta)$ within a solid element e, the quadratic system

$$\mathbf{x}_{\alpha,\beta} = \sum_j \mathbf{x}_j^e N_j^e(\xi,\eta) \tag{22}$$

can be solved for the reference element coordinate (ξ,η) by Newton's method. The body force $\mathbf{B}$ at $\mathbf{x}_{\alpha,\beta}$ is then interpolated as

$$\mathbf{B}_{\alpha,\beta} = \sum_j \mathbf{B}_j^e N_j^e(\xi,\eta). \tag{23}$$

3.3.2. Immersed-interface method

The immersed-interface method, as proposed for membranes [29,53], is formulated to minimize the truncation error of the discrete momentum equation at every Cartesian mesh point. Away from the interface, (14) applies with $\mathbf{F} = 0$ and $\mathbf{B} = 0$ in the fluid. Within the solid, the body force $\mathbf{B}$ is again interpolated pointwise as discussed in Section 3.3.1. The singular $\mathbf{F}$ is analogous to a membrane force on a fluid. To incorporate this, the regular finite-difference stencils are modified to respect the interface jump conditions (5a) and (13). For example, if p and its derivative p_x are discontinuous at $x_{i+\alpha}$ ($\alpha > 1/2$) with jumps

$$[\![p]\!] = p(x_{i+\alpha}^+) - p(x_{i+\alpha}^-), \tag{24a}$$

$$[\![p_x]\!] = p_x(x_{i+\alpha}^+) - p_x(x_{i+\alpha}^-), \tag{24b}$$

then a consistent modified finite-difference stencil for $(p_x)_{i+\frac{1}{2}}$ is

$$\left(\frac{\partial p}{\partial x}\right)_{i+\frac{1}{2}} = \frac{p_{i+1} - p_i - [\![p]\!]}{h} - (1-\alpha)[\![p_x]\!] + \mathrm{O}(h). \tag{25}$$

Arbitrary order finite-difference stencils are given by Xu and Wang [53].

For our fluid–structure solver, the jumps in $\mathbf{u}$, p and their derivatives are needed to modify the finite-difference stencils, and they can be derived from (14) and (3b). In two dimensions, the fluid–solid interface Γ is one-dimensional, so to achieve global second-order spatial accuracy, $\mathrm{O}(h)$ local truncation errors are required for the modified finite-difference stencils that span Γ [30,36]. This requires jump conditions up through second-order derivatives of velocity and first-order derivatives of pressure. Because of the moving interface, the temporal derivatives are also discontinuous when Γ passes through an Eulerian mesh point, thus a second-order accurate time integration requires jumps in time derivative as well as mixed space–time derivatives through second order [32,53]. For now, our primary interest is the general investigation of fixed-mesh algorithms, so the first order, easy-to-implement immerse-interface method by Lee and LeVeque [29] is adopted in this study. Extension to higher order schemes is possible [53,56].

For our system, the jump conditions for velocity and pressure are [55]

$$[\![p]\!] = \hat{f}_\mathrm{n}, \quad \left[\!\!\left[\frac{\partial p}{\partial n}\right]\!\!\right] = \frac{\partial \hat{f}_\mathrm{t}}{\partial s} + [\![B_\mathrm{n}]\!], \tag{26a}$$

$$[\![\mathbf{u}]\!] = 0, \quad \mu_\mathrm{f}\left[\!\!\left[\frac{\partial \mathbf{u}}{\partial n}\right]\!\!\right] = -\hat{f}_\mathrm{t}\mathbf{t}, \tag{26b}$$

where $\hat{\mathbf{f}} = -\boldsymbol{\tau}_{\text{elas}} \cdot \mathbf{n}$ is the surface force density (17) with tangential component $\hat{f}_{\text{t}}$ and normal component $\hat{f}_{\text{n}}$, and $[\![B_{\text{n}}]\!]$ is the jump in the normal component of the body force (15)

$$[\![B_{\text{n}}]\!] = \mathbf{B} \cdot \mathbf{n}|_{\text{fluid}} - \mathbf{B} \cdot \mathbf{n}|_{\text{solid}} = -(\nabla \cdot \boldsymbol{\tau}_{\text{elas}}) \cdot \mathbf{n}. \tag{27}$$

It is clear that $\hat{f}_{\text{t}}$ causes the discontinuity in velocity gradient while $\hat{f}_{\text{n}}$ leads to the jump in pressure. Following Lee and Leveque [29], the tangential force $\hat{f}_{\text{t}}\mathbf{t}$ is spread to the Cartesian mesh by the standard immersed-boundary method as discussed in Section 3.3.1, while the $\hat{f}_{\text{n}}\mathbf{n}$ part in the jump conditions is retained. Without $\hat{f}_{\text{t}}$, the jump conditions are simplified:

$$[\![p]\!] = \hat{f}_{\text{n}}, \quad \left[\!\!\left[\frac{\partial p}{\partial n} \right]\!\!\right] = [\![B_{\text{n}}]\!], \tag{28a}$$

$$[\![\mathbf{u}]\!] = 0, \qquad \mu_{\text{f}} \left[\!\!\left[\frac{\partial \mathbf{u}}{\partial n} \right]\!\!\right] = 0. \tag{28b}$$

The spreading of the tangential surface force smears the discontinuity in velocity gradient, so the velocity solution is formally first-order accurate in space, but in practice this nominally low order is unimportant, at least for our demonstrations. The discontinuities in pressure and its gradient are, however, preserved. With ∇p being the only term in (14) whose finite-difference stencils need to be modified, the numerical implementation is relatively easy and is done exactly as in [29,55].

3.3.3. Force-projection method

This last approach adopts the central idea of the immersed-boundary method whereby the surface force is approximated by a continuous body-force distribution, but as opposed to the immersed-boundary method discussed in Section 3.3.1, $\mathbf{F}$ and $\mathbf{B}$ are distributed in a single framework that exactly conserves momentum. We also seek to avoid the convolution with *ad hoc* kernel functions like (20) and reduce smearing at the interface.

Given a continuous function q, a natural approximation to q is the piecewise bilinear interpolation based on its values at points on a Cartesian mesh:

$$q(\mathbf{x}) \approx \sum_{ij} \psi_{ij}(\mathbf{x}) q_{ij}, \tag{29}$$

where q_{ij} is the value of q at mesh point (i, j) and ψ_{ij} is the associated piecewise bilinear shape function. For a point $\mathbf{x} = (x, y)$ located in a cell $[x_i, y_j] \times [x_{i+1}, y_{j+1}]$ with

$$x = x_i + s_x h_x, \quad y = y_j + s_y h_y \quad (0 \leqslant s_x, s_y \leqslant 1), \tag{30}$$

the bilinear interpolation at $\mathbf{x}$ is

$$q(\mathbf{x}) = \psi_{i,j}(\mathbf{x}) q_{i,j} + \psi_{i+1,j}(\mathbf{x}) q_{i+1,j} + \psi_{i+1,j+1}(\mathbf{x}) q_{i+1,j+1} + \psi_{i,j+1}(\mathbf{x}) q_{i,j+1}, \tag{31}$$

where

$$\psi_{i,j}(\mathbf{x}) = (1 - s_x)(1 - s_y), \tag{32a}$$

$$\psi_{i+1,j}(\mathbf{x}) = s_x(1 - s_y), \tag{32b}$$

$$\psi_{i+1,j+1}(\mathbf{x}) = s_x s_y, \tag{32c}$$

$$\psi_{i,j+1}(\mathbf{x}) = (1 - s_x) s_y. \tag{32d}$$

When the solid is completely submerged within the fluid so that $\partial\Omega_{\text{s}} = \Gamma$, the body force $\mathbf{B}$ and the surface force $\mathbf{F}$ can be combined as

$$\mathbf{B} + \mathbf{F} = \nabla \cdot (\chi_{\text{s}} \boldsymbol{\tau}_{\text{elas}}), \tag{33}$$

which follows from the analysis in Section 2, with weak derivatives defined in the usual way [37], and an application of Green's theorem. Complete details are provided in [55]. Eq. (33) motivates the approximation

$$\nabla \cdot (\chi_{\text{s}} \boldsymbol{\tau}_{\text{elas}}) \approx \sum_{ij} \psi_{ij}(\mathbf{x}) \mathbf{f}_{ij}, \tag{34}$$

where $\mathbf{f}_{ij}$ is the distributed force at each Cartesian mesh point. The application of a standard Galerkin projection leads to

$$\sum_{\beta}\left(\int_{\Omega}\psi_{\alpha}\psi_{\beta}\,\mathrm{d}\mathbf{x}\mathbf{f}_{\beta}\right)=-\int_{\Omega_s}\nabla\psi_{\alpha}\cdot\boldsymbol{\tau}_{\mathrm{elas}}\,\mathrm{d}\mathbf{x} \tag{35}$$

for any Cartesian mesh point α. By lumping the stiffness matrix coefficients, $\mathbf{f}_{ij}$ at any Cartesian mesh point can be explicitly computed as

$$\mathbf{f}_{ij}=-\frac{1}{h_x h_y}\int_{\Omega_s}\nabla\psi_{ij}\cdot\boldsymbol{\tau}_{\mathrm{elas}}\,\mathrm{d}\mathbf{x}. \tag{36}$$

This single, self-consistent formulation for **B** and **F** will give the combined formulation exact conservation of momentum as shown in Section 3.3.4.

Because of the mismatch between the Cartesian mesh and the Lagrangian mesh, the numerical integration of $\nabla\psi_{ij}\cdot\boldsymbol{\tau}_{\mathrm{elas}}$ in (36) requires special consideration. Each ψ_{ij} is piecewise bilinear on the Cartesian mesh, and its gradient is discontinues across Cartesian mesh cells. On the other hand, $\boldsymbol{\tau}_{\mathrm{elas}}$ is piecewise smooth on the Lagrangian mesh. The integrand is thus in general not smooth in any Cartesian mesh cell or any Lagrangian mesh element, so standard quadrature will not be accurate. An apparent solution is a collocation approximation that is similar to the scheme used by the material-point method [46,47]. We first divide every solid mesh element e into subelements. The division is uniform on the reference element, as shown in Fig. 3. The stress distribution within e can then be approximated by

$$\boldsymbol{\tau}_{\mathrm{elas}}(\mathbf{x})=\int_{e}\boldsymbol{\tau}_{\mathrm{elas}}(\mathbf{y})\delta(\mathbf{x}-\mathbf{y})\,\mathrm{d}\mathbf{y}\approx\sum_{e'}\boldsymbol{\tau}_{\mathrm{elas}}(\mathbf{x}_{e'})\delta(\mathbf{x}-\mathbf{x}_{e'})A_{e'}, \tag{37}$$

where $\mathbf{x}_{e'}$ is the collocation point within subelement e', and $A_{e'}$ is the area of e'. Each $\mathbf{x}_{e'}$ is like a material point upon which the elastic stress in e' is concentrated. The integration in (36) over element e is simply

$$\int_{e}\nabla\psi_{ij}\cdot\boldsymbol{\tau}_{\mathrm{elas}}\,\mathrm{d}\mathbf{x}\approx\sum_{e'}\nabla\psi_{ij}(\mathbf{x}_{e'})\boldsymbol{\tau}_{\mathrm{elas}}(\mathbf{x}_{e'})A_{e'}. \tag{38}$$

This collocation approach, although easy to implement, is shown to be unstable when used for load transfer in aero-elasticity simulations [22]. Also, since the subelements are still not consistent with the Cartesian mesh cells, the size of each subelement needs to be $O(h^2)$ for the integration error in $\mathbf{f}_{ij}$ to be $O(h)$, where h is the Cartesian mesh spacing. The computational cost thus would be prohibitively high to satisfy such a requirement. For our simulations, spatial oscillations in streamlines within the solid are observed when (38) is used.

We instead evaluate the integration (36) using a common refinement approach. Every Lagrangian mesh element e is divided into subelements, every one of which is confined within one single Cartesian mesh cell. The integrand of (36) is then smooth within every subelement upon which the integration is computed. Because of the simple structure of the Cartesian mesh, the common refinement is done in an easier and faster way compared to that for the general case [23]. We first divide the original element e into triangles as in the collocation

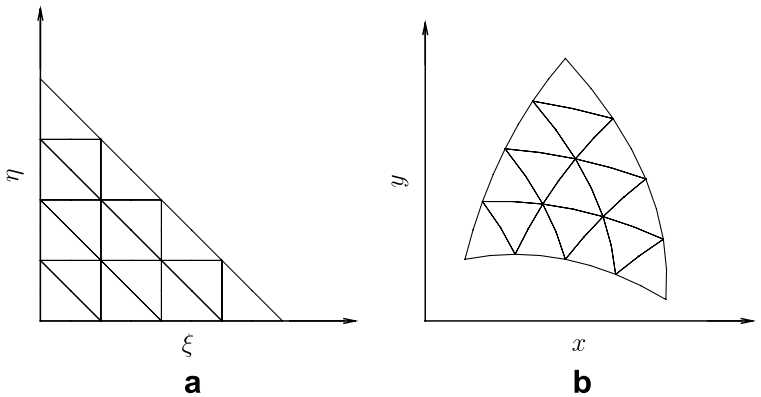


Fig. 3. Uniform subdivision of a Lagrangian mesh element in (a) reference coordinate space and (b) physical space.

method. If e is a triangular element, it is divided into 16 smaller triangles; if e is quadrilateral, it is divided into 32 triangles. We treat each sub-triangle as a linear triangle, and use the Cartesian mesh lines to cut through this sub-triangle to form a collection of polygons whose sides are either parts of the sub-triangle's boundary or coincide with the Cartesian mesh lines, as shown in Fig. 4(a). The initial sub-triangle and any intermediate polygon have straight edges, making it trivial to find their intersections with the Cartesian mesh lines, which is the only operation needed that is tied to the geometry of the two meshes. Hence the common refinement invokes minimal computational cost.

The number of edges of each polygon divided ranges between three and seven. The polygon could be further tessellated into triangles and quadrilaterals upon which the standard numerical quadrature can be used to compute the integral in (36). However, since linear approximation is used on every sub-triangle, the elastic stress $\boldsymbol{\tau}_{\text{elas}}$ is also linear about x and y over every divided polygon. Likewise, because of the common refinement, the gradient of ψ_{ij} is also linear in x and y on each polygon. The integrand $\nabla\psi_{ij}\cdot\boldsymbol{\tau}_{\text{elas}}$ is thus quadratic within each polygon, with each component having the form $\sum_{mn}a_{mn}x^m y^n$, where $m+n\leqslant 2$ and the coefficients a_{mn} can be analytically computed. The integration of this polynomial over the interior of an arbitrary polygon P can be converted to an integral along the edges of P by Green's theorem [6],

$$\int_P \sum_{mn} a_{mn}x^m y^n \,\mathrm{d}x\,\mathrm{d}y = \int_P \sum_{mn} \frac{\partial}{\partial x}\left(\frac{a_{mn}}{m+1}x^{m+1}y^n\right)\mathrm{d}x\,\mathrm{d}y = \int_{\partial P}\sum_{mn}\frac{a_{mn}}{m+1}x^{m+1}y^n n_x\,\mathrm{d}l, \tag{39}$$

where n_x is the x component of the boundary's outward normal. Along an edge of the polygon, n_x is constant, and x and y are both linear functions of the edge length l. The integrand in the last integral of (39) is thus a piecewise cubic function of l, and the integration on every edge can be exactly calculated by the two-point Gauss quadrature rule, as in Fig. 4(b). The only numerical error introduced is the linear approximation made to the intermediate sub-triangles. The interpolation is thus formally second-order accurate with Lagrangian mesh size, but for well-resolved solid deformation, this error is negligible compared to the errors of other numerical components. Compared to the collocation-point method, this common refinement approach retains higher order of accuracy, is stable, and is still fairly easy to implement.

As formulated, the force-projection method (36) applies only when the solid is completely submerged such that $\partial\Omega_{\text{s}} = \Gamma$, but it can be easily generalized to cases where only part of the solid boundary is a fluid–solid interface, while the other part might be, for example, a Dirichlet boundary. In that case, $\partial\Omega_{\text{s}}$ is divided such that $\partial\Omega_{\text{s}} = \Gamma\cup D$, as shown in Fig. 5, which upon following the reasoning that leads to (33) yields

$$\nabla\cdot(\chi_{\text{s}}\boldsymbol{\tau}_{\text{elas}}) = \mathbf{F}+\mathbf{B}+\mathbf{F}_D, \tag{40}$$

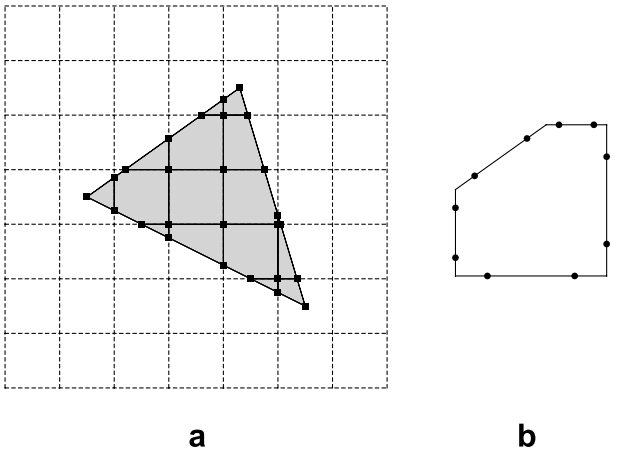


Fig. 4. Common refinement between the triangle and Cartesian meshes: (a) the polygons generated by finding vertices ■ as the intersections of the straight lines making up the two meshes; (b) ● the quadrature points for exactly evaluating the force-projection integral (36) over one of the polygons.

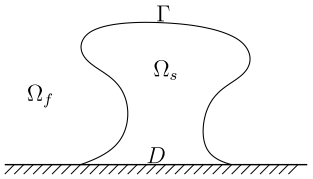


Fig. 5. Division of a solid boundary into fluid–solid interface Γ and Dirichlet boundary D.

where $\mathbf{F}_D$ is a surface force distribution on D defined in the same way as $\mathbf{F}$,

$$\mathbf{F}_D(\mathbf{x},t) = \int_D \hat{\mathbf{f}}(s,t)\,\delta(\mathbf{x}-\mathbf{r}(s,t))\,\mathrm{d}s, \tag{41}$$

with $\hat{\mathbf{f}} = -\boldsymbol{\tau}_{\text{elas}}\cdot\mathbf{n}$ as before. The force-projection formula (36) is then modified so that $\mathbf{F}_D$ is excluded from $\nabla\cdot(\chi_s\boldsymbol{\tau}_{\text{elas}})$, and

$$\mathbf{f}_{ij} = \frac{1}{h_x h_y}\left(-\int_{\Omega_s}\nabla\psi_{ij}\cdot\boldsymbol{\tau}_{\text{elas}}\,\mathrm{d}\mathbf{x} + \int_D \psi_{ij}\boldsymbol{\tau}_{\text{elas}}\cdot\mathbf{n}\,\mathrm{d}s\right), \tag{42}$$

which can be evaluated as before.

3.3.4. Discussion on the three force-distribution approaches

In the previous sections, three strategies were proposed to transfer the solid elastic forces to the Cartesian mesh. In both the immersed-boundary and force-projection methods, the original force distribution is approximated by continuous body forces, which are directly added to the discrete Navier–Stokes momentum equation. On the other hand, the immersed-interface method uses those forces to modify the finite-difference stencils near the interface to minimize local truncation errors. In this sense, the force distributions in the immersed-boundary and force-projection methods are more like finite-volume approaches, while the immersed-interface method is purely finite difference.

The different strategies affect the accuracy of the velocity and pressure solutions. The approximation of replacing the surface force by a narrow-banded body force, as in the immersed-boundary and force-projection methods, is made at the level of the governing equations. The width of the band is $4h$ for the immersed-boundary method using the kernel function (20) and $2h$ for the force-projection method. The sharp jumps of velocity and pressure are smeared to similar width, so the spatial accuracy in an L_∞ norm is first order at best. On the other hand, the immersed-interface method incurs only local truncation error in the final discretized governing equations, so the order of accuracy achievable, in principle, depends only on the highest order of spatial and temporal derivatives whose jump conditions can be computed. The preserved sharp interface in the immersed-interface method also results in higher (sub-grid) spatial resolution at the interface, which is especially important when there are flow boundary layers at the interface.

There are, however, some difficulties with the immersed-interface method:

- High-order jump terms are sensitive to high wave number perturbations at the interface, and they also make the method more susceptible to aliasing errors [10]. Therefore, filtering of quantities on Γ is often needed. Filtering without adding excessive dissipation can be difficult for two-dimensional interfaces.
- To decouple the solution of the velocity and pressure, a split-step method is commonly used in time integration such as the one used here (see Section 3.5). The numerical pressure (or pressure increment) is typically solved by a Poisson equation, and is a first- or second-order time-accurate approximation to the analytical pressure in a pure fluid system. On the other hand, the jump conditions are derived for the analytical pressure. The implications of this inconsistency do not appear to have been analyzed.
- For more general problems, the fluid and solid have mismatched densities and viscosities. In this case, the jumps in velocity and pressure will not depend locally on the interfacial force density and geometry. Take a membrane/Stokes-flow system as an example: when the fluids on the two sides of the membrane have

different viscosities, the discontinuity in velocity gradient can only be solved exactly from an implicit boundary integral equation [42]. Any local jump condition is essentially an approximation to this, with consequences that are difficult to assess.

Momentum conservation is another important issue. For a completely submerged solid body, the forces **B** and **F** together should have zero contribution to the total linear momentum of the system. This can be seen from

$$\int_{\Omega} (\mathbf{B} + \mathbf{F})\,\mathrm{d}\mathbf{x} = \int_{\Omega_s} \nabla \cdot \boldsymbol{\tau}_{\mathrm{elas}}\,\mathrm{d}\mathbf{x} - \int_{\Gamma} \boldsymbol{\tau}_{\mathrm{elas}} \cdot \mathbf{n}\,\mathrm{d}S = 0, \tag{43}$$

where we have used the Green's theorem and the fact that $\Gamma = \partial\Omega_s$. In order to discretely conserve linear momentum, the forces transferred to the Cartesian mesh should have zero sum. However, the sequential computation of $\boldsymbol{\tau}_{\mathrm{elas}}$ and $\nabla \cdot \boldsymbol{\tau}_{\mathrm{elas}}$ in the immersed-boundary and immersed-interface approaches does not guarantee (43) in the presence of finite numerical errors, so the transferred force on the Cartesian mesh will not in general conserve momentum. On the other hand, the force-projection formula (36) does conserve momentum independently of the discretization's details. This can be shown by summing $\mathbf{f}_{ij}$ in (36) over the whole domain and using the fact that $\sum_{ij} \nabla\psi_{ij}(\mathbf{x}) = 0$ for any $\mathbf{x}$ [55].

3.4. Solid tracking and the incompressibility constraint

Solid incompressibility requires that $\det \mathbf{A} = 1$. For a material point, the local volume expansion $\det \mathbf{A}$ changes with time as

$$\frac{1}{\det \mathbf{A}} \frac{\mathrm{D} \det \mathbf{A}}{\mathrm{D}t} = \nabla \cdot \mathbf{u}, \tag{44}$$

so the incompressibility constraints defined in Eulerian and Lagrangian frames are, of course, analytically compatible. In our projection method for the Eulerian solver (Section 3.1), the velocity is constructed to be divergence free, but this compatibility is disrupted when the velocities are numerically interpolated from the Cartesian mesh to move the solid. Since the solid constitutive law does not have any resistance to local volume changes, no feedback is provided to suppress this deviation as it inevitably occurs. Without any special treatment, we observe that $\|\det \mathbf{A} - 1\|$ does indeed increase slowly with time, eventually leading to instabilities.

Artificial compressibility offers a crude means of correcting this, but restricts the time step by introducing fast compressive modes. Ideally, the solid incompressibility should be enforced as a constraint by a pressure-like Lagrange multiplier in Ω_s [21]. The role of this constraint is functionally redundant with the global pressure field already determined from the velocity divergence-free constraint, so it should only be required to make small corrections at each time step and can therefore be implemented simply as follows. During every time step, the Lagrangian mesh points are first moved without constraints at the interpolated velocities. We then seek a correction $\mathbf{g}$ such that

$$1 = \det \frac{\partial(\mathbf{x} + \mathbf{g})}{\partial \mathbf{X}}. \tag{45}$$

The anticipated smallness of $\mathbf{g}$ suggests the linear approximation

$$1 \approx \det \mathbf{A} + \frac{\partial \det \mathbf{A}}{\partial A_{ij}} \frac{\partial g_i}{\partial X_j} = \det \mathbf{A} + \det \mathbf{A} A^{-1}_{ji} \frac{\partial g_i}{\partial X_j}. \tag{46}$$

Since

$$A^{-1}_{ji} \frac{\partial g_i}{\partial X_j} = \frac{\partial X_j}{\partial x_i} \frac{\partial g_i}{\partial X_j} = \frac{\partial g_i}{\partial x_i}, \tag{47}$$

$\mathbf{g}$ can be regarded as a function of the current solid coordinates that satisfies

$$\frac{\partial g_i}{\partial x_i} - \det \mathbf{A}^{-1} + 1 = 0. \tag{48}$$

This specifies a manifold of all admissible $\mathbf{g}$'s, from which we seek the one with the smallest L_2-norm. This constrained minimization problem is solved approximately (but accurately for our applications) by a penalty method. This amounts to minimizing the functional

$$I = \sum_{\alpha} \frac{|\mathbf{g}_a|^2}{2A_\alpha} + \sum_e \frac{1}{2A_e} \int_e (\nabla \cdot \mathbf{g} - \det \mathbf{A}^{-1} + 1)^2 \, \mathrm{d}\mathbf{x}, \tag{49}$$

where α is the mesh point index, A_α is the sum of the area of all mesh elements surrounding point α, e is the mesh element index, and A_e is the area of e. The elementwise integrations are scaled by the mesh element area, so no mesh dependent penalty parameters are needed. The term $\mathbf{g}$ is interpolated isoparametrically, and when discretized, I is quadratic in the nodal values of $\mathbf{g}$:

$$I = \sum_{\alpha} \frac{|\mathbf{g}_\alpha|^2}{2A_\alpha} + \sum_e \frac{1}{2A_e} \int_e \left(\sum_l \nabla N_l^e \cdot \mathbf{g}_l^e - \det \mathbf{A}^{-1} + 1 \right)^2 \mathrm{d}\mathbf{x}, \tag{50}$$

where l is the local nodal index within element e. Taking the derivative of I with respect to every $\mathbf{g}_\alpha$ results in a symmetric positive-definite linear system that requires

$$\frac{\mathbf{g}_\alpha}{A_\alpha} + \sum_e \frac{1}{A_e} \int_e \sum_l \nabla N_\alpha (\nabla N_l^e \cdot \mathbf{g}_l^e) \, \mathrm{d}\mathbf{x} = \sum_e \frac{1}{A_e} \int_e \nabla N_\alpha (\det \mathbf{A}^{-1} - 1) \, \mathrm{d}\mathbf{x} \tag{51}$$

for every α. This linear system is solved for the $\mathbf{g}$'s by a standard conjugate-gradient algorithm.

3.5. Time integration

The full system is integrated in time with a hybrid third-order Runge–Kutta/Crank–Nicolson scheme, which is implicit in the viscous term [45] and uses a projection method to compute p to enforce $\nabla \cdot \mathbf{u} = 0$. To simplify notation, we define

$$\mathbf{R}_\mathrm{f} \equiv \mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \mu_\mathrm{f} \nabla^2 \mathbf{u} \tag{52}$$

and let $\mathbf{S} \equiv \mathbf{F} + \mathbf{B}$ be the total elastic force.

To decouple the solution of velocity, pressure and solid displacement, in each substep j of the time scheme, we first update the solid Lagrangian mesh, then compute $\mathbf{S}$ from the updated solid displacements, and finally solve the velocity and pressure:

$$\frac{\mathbf{x}^{(j+1)} - \mathbf{x}^{(j)}}{\Delta t} = \gamma_{j+1} \mathbf{u}^{(j)}(\mathbf{x}^{(j)}) + \zeta_j \mathbf{u}^{(j-1)}(\mathbf{x}^{(j-1)}), \tag{53a}$$

$$\frac{\tilde{\mathbf{u}}^{(j+1)} - \mathbf{u}^{(j)}}{\Delta t} = \mathbf{R}_\mathrm{f}^{(j+1)} + (\alpha_{j+1} \mathbf{S}^{(j)} + \beta_{j+1} \mathbf{S}^{(j+1)}), \tag{53b}$$

where

$$\mathbf{R}_\mathrm{f}^{(j+1)} = -(\gamma_{j+1} \mathbf{u}^{(j)} \cdot \nabla \mathbf{u}^{(j)} + \zeta_j \mathbf{u}^{(j-1)} \cdot \nabla \mathbf{u}^{(j-1)}) - (\alpha_{j+1} + \beta_{j+1}) \nabla p^{(j)} + (\alpha_{j+1} \nabla^2 \mathbf{u}^{(j)} + \beta_{j+1} \nabla^2 \tilde{\mathbf{u}}^{(j+1)}), \tag{53c}$$

$$\nabla^2 \phi = \nabla \cdot \tilde{\mathbf{u}}^{(j+1)}, \tag{53d}$$

$$\mathbf{u}^{(j+1)} = \tilde{\mathbf{u}}^{(j+1)} - \nabla \phi, \tag{53e}$$

$$p^{(j+1)} = p^{(j)} + \frac{\phi}{(\alpha_{j+1} + \beta_{j+1}) \Delta t}. \tag{53f}$$

Here, $j = 0, 1, 2$, so $q^{(0)} = q^n$ indicates a quantity at the beginning of a time step, and $q^{(3)} = q^{n+1}$ at the time step's end, with the exception of $\mathbf{x}^{(3)}$ which must be corrected for solid incompressibility to yield $\mathbf{x}^{n+1} = \mathbf{x}^{(3)} + \mathbf{g}(\mathbf{x}^{(3)})$. The parameters used in (53) are [45]

$$\gamma_1 = \frac{8}{15}, \qquad \gamma_2 = \frac{5}{12}, \qquad \gamma_3 = \frac{3}{4}, \tag{54a}$$

$$\zeta_0 = 0, \qquad \zeta_1 = -\frac{17}{60}, \qquad \zeta_2 = -\frac{5}{12}, \tag{54b}$$

$$\alpha_1 = \frac{29}{96}, \qquad \alpha_2 = -\frac{3}{40}, \qquad \alpha_3 = \frac{1}{6}, \tag{54c}$$

$$\beta_1 = \frac{37}{160}, \qquad \beta_2 = \frac{5}{24}, \qquad \beta_3 = \frac{1}{6}. \tag{54d}$$

A Courant stability limit for the convection and solid elastic force terms restricts the time step. There are two propagation velocities that must be considered: the convection velocity and the elastic wave speed $\sqrt{\mu_s/\rho}$. To estimate the maximum stable time step, a composite Courant stability constraint is defined

$$\mathrm{CFL} = \Delta t \max\left(\frac{|u|}{h_x} + \frac{|v|}{h_y} + \frac{\lambda_c}{h_s}\sqrt{\frac{\tilde{\mu}_s}{\rho}}\right) \leqslant \sqrt{3}, \tag{55}$$

where $\tilde{\mu}_s$ is zero in the fluid and has value μ_s in the solid, h_s is the grid parameter describing the Lagrangian grid size and is defined as the smallest side length of all mesh elements, and the $\sqrt{3}$ limit arises from the stability bound of the third-order Runge–Kutta scheme. The coefficient $\lambda_c = 3.0$ is an empirically estimated coefficient for the wave speeds on the unstructured mesh. It was selected by considering the modulus of the maximum eigenvalues of the stiffness matrix, linearized about the undeformed solid configuration, for a number of solid grids and geometries, and requiring that $\lambda_c/h_s\sqrt{\mu_s/\rho}$ be at least 50% larger than the observed eigenvalues [55]. This estimate of maximum stable time step is consistent with our experience with numerical tests.

4. Numerical results

4.1. Convergence

The order of accuracy of the combined scheme can be estimated by considering its components. The ZZ patch recovery method for computing the elastic stress is third-order accurate with respect to the Lagrangian mesh element size, and the finite differences on the Cartesian mesh are second-order accurate with respect to the Cartesian mesh spacing. However, the accuracy is reduced for a fluid–solid system since the elastic force **F** in (14) from the Lagrangian mesh is spread to nearby Cartesian mesh points, which is well known to yield a velocity solution that is at most first-order accurate in an L_∞ norm [27]. This loss of accuracy also occurs for the immersed-interface approach because of the diffusion of the tangential component of **F**. The time integration scheme in Section 3.5 is formally second-order accurate, but only for solutions smooth in time. When the fluid–solid interface passes a Cartesian mesh point, the time derivatives of the velocity and pressure at this point are also discontinuous. Without any correction to account for this temporal non-smoothness, the temporal accuracy is also formally of first order in L_∞ [32,53].

Two model systems are simulated to confirm the accuracy of the method. We first consider the motion of a viscoelastic solid without a fluid–solid interface in Section 4.1.1 and then two fluid–solid interaction systems in Sections 4.1.2 and 4.1.3. For every system simulated, the convergence with respect to the solid Lagrangian mesh element size and the Cartesian mesh spacing are evaluated separately. Unless specified, the transfer of the elastic forces is by the new force-projection formula (36).

4.1.1. A pure solid system

The simulation of the motion of a pure solid exercises all the components of the scheme except the fluid–solid interface, which is more challenging to analyze. The solid is initially stress free, and occupies a square domain $[-0.5, 0.5] \times [-0.5, 0.5]$. It is then deformed by a velocity field with stream function

$$\psi(\mathbf{x}, t) = \psi_0 \sin(\omega t)\sin(k_x x)\sin(k_y y), \tag{56}$$

where $\phi_0 = 0.25$, $\omega = 2\pi$ and $k_x = k_y = 2\pi$. The viscosity and the elastic constant are $\mu_f = 0.1$ and $\mu_s = 1$. The solid deformation at $t = 0.5$ is shown in Fig. 6.

The algorithm is then tested by applying the external body force $\mathbf{b}_{ex}$ that would generate the target velocity field in the numerical solution. It is

$$\mathbf{b}_{ex} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \mu_f \nabla^2 \mathbf{u} - \nabla \cdot \boldsymbol{\tau}_{elas}, \tag{57}$$

where the pressure has been set to be zero. All right-hand side terms in (57) that depend upon $\mathbf{u}$ can be evaluated analytically. To compute $\nabla \cdot \boldsymbol{\tau}_{elas}$, the initial position $\mathbf{X}$ for every Cartesian mesh point is found by integrating (4) backward in time using a fourth-order Runge–Kutta scheme. The time step is chosen such that halving the time step changes the computed $\mathbf{X}$ by less than 1 part in 10^6. The inverse deformation tensor $\mathbf{A}^{-1}$, which gives $\boldsymbol{\tau}_{elas}$ and $\nabla \cdot \boldsymbol{\tau}_{elas}$, are computed with Fourier methods.

For these tests, the undeformed solid is divided into uniform square elements. The errors in the calculated velocity and solid displacement are computed at $t = 0.5$. For the finest mesh, changing the time step from 1/1024 to 1/2048 changes the error by less than 1%, so the temporal error is negligible compared to the spatial errors we are assessing.

Convergence with Lagrangian mesh refinement is shown in Fig. 7, where a 512×512 Cartesian mesh is used, and the number of the Lagrangian mesh elements, N_s, increases from 4×4 to 64×64. Every Lagrangian mesh element has relaxed side length $h_s = N_s^{-\frac{1}{2}}$. The error in the velocity and solid displacement in all three norms shown (L_1, L_2 and L_∞) decreases as $O(h_s^3)$ until N_s exceeds 32×32, after which point the error introduced by the ZZ patch recovery is comparable with the Cartesian-mesh Navier–Stokes solver. Convergence with respect to the Cartesian mesh spacing h is shown in Fig. 8, where the Lagrangian mesh is 64×64, and the Cartesian mesh is refined from 4×4 to 256×256. The error in velocity decreases as $O(h^2)$ in all norms; the error in solid displacement decreases as $O(h^2)$ in the L_∞ norm, and at a slightly slower rate $O(h^{1.9})$ in the L_1 and L_2 norms. Possible reasons for this slight anomaly are discussed in Section 4.1.4.

In summary, the elastic stress computed on the Lagrangian mesh is confirmed to be third-order accurate with respect to the Lagrangian mesh element size; in the absence of the fluid–solid interface, the elastic force transferred to the Cartesian mesh is second-order accurate with respect to the Cartesian mesh spacing.

4.1.2. The deformation of an elastic wall by flow

In this test, we simulate the deformation of a soft wall by fluid flow, in a lid-driven cavity flow as reported by Dunne [8]. The cavity is 2 by 2, with bottom 0.5 occupied by a neo-Hookean wall and the upper part filled with fluid. The velocities are zero at all cavity boundaries except the top lid, where the x component of the velocity is

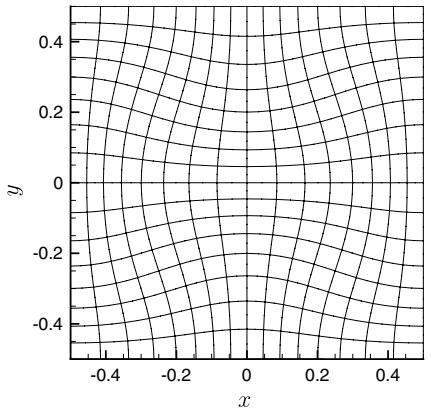
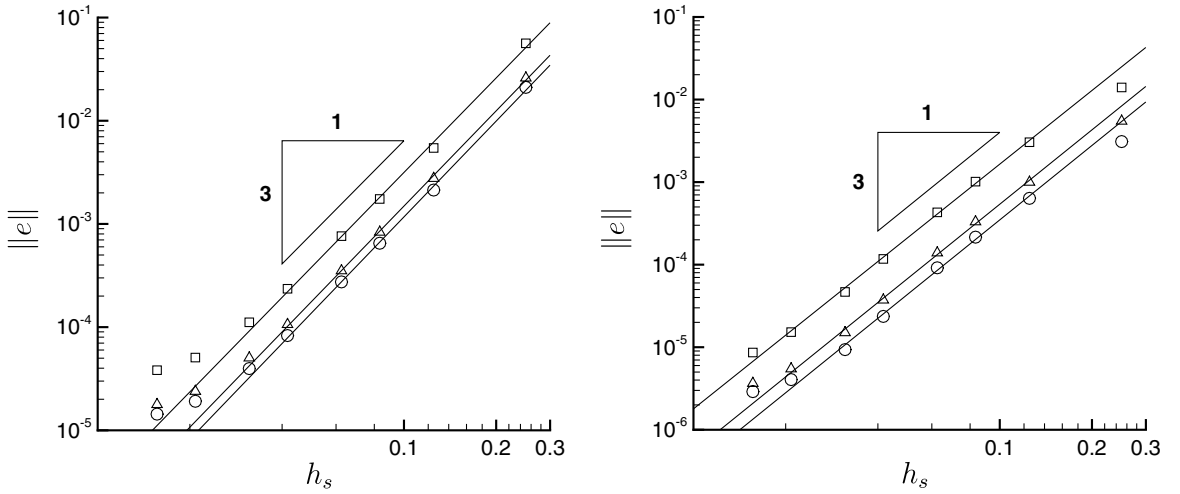
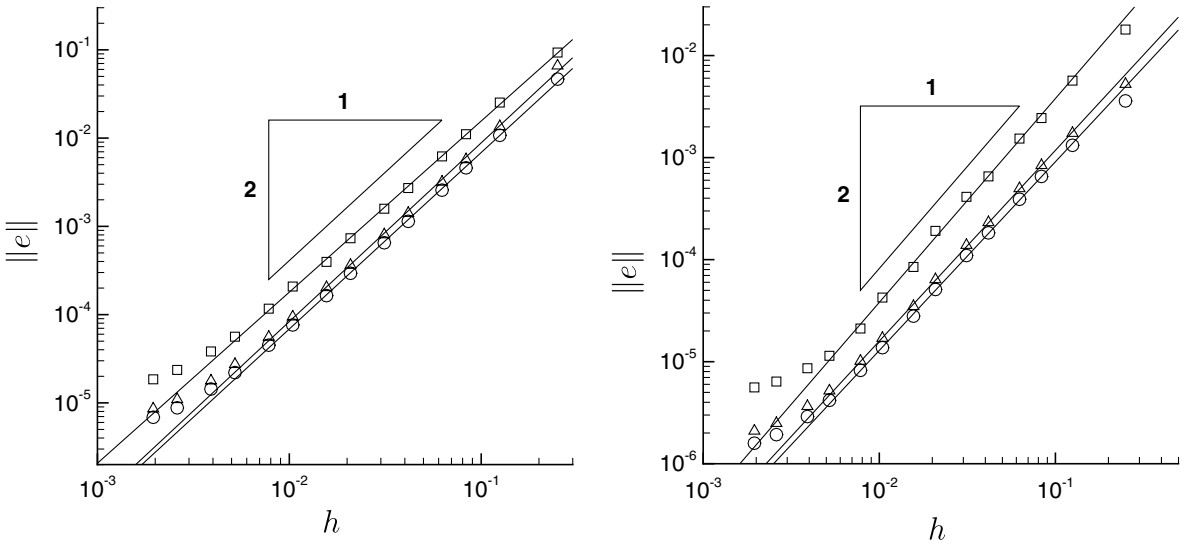


Fig. 6. The deformed solid mesh in the solid-only convergence test.



(a) Error in velocity field (b) Error in solid displacement

Fig. 7. Pure solid system: convergence with the Lagrangian mesh element size. The underlying Cartesian mesh has 512×512 mesh cells. Plotted is the norm of the numerical error e: ○ L_1, △ L_2, and □ L_∞.



(a) Error in velocity field (b) Error in solid displacement

Fig. 8. Pure solid system: convergence with the Cartesian mesh spacing. The solid Lagrangian mesh has 64×64 elements. Plotted is the norm of the numerical error e: ○ L_1, △ L_2, and □ L_∞.

$$u = 0.5\begin{cases} \sin^2(\pi x/0.6), & x \in [0.0, 0.3], \\ 1, & x \in (0.3, 1.7), \\ \sin^2(\pi(x-2.0)/0.6) & x \in [1.7, 2.0]. \end{cases} \tag{58}$$

The fluid viscosity is $\mu_f = 0.2$, and the solid has $\mu_s = 0.2$. The convection term in the Navier–Stokes equation is omitted as in [8]. The cavity is discretized by a uniform 256×256 Cartesian mesh, and the elastic wall by a 16×4 quadrilateral mesh. The simulation runs until $\|\mathbf{u}\|_\infty < 10^{-4}$ in Ω_s, when the system is considered to have reached a steady sate. The wall deformation and the streamlines are shown in Fig. 9 to compare well with

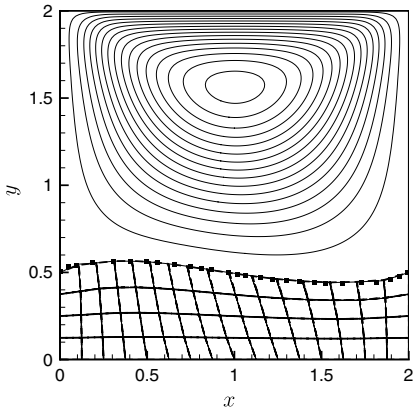


Fig. 9. The steady state of the deformation of an elastic wall driven by flow. The ■ shows the interface position of Dunne [8]. The solid mesh is shown both for the present force-projection method — and an Arbitrary Lagrangian–Eulerian method ---.

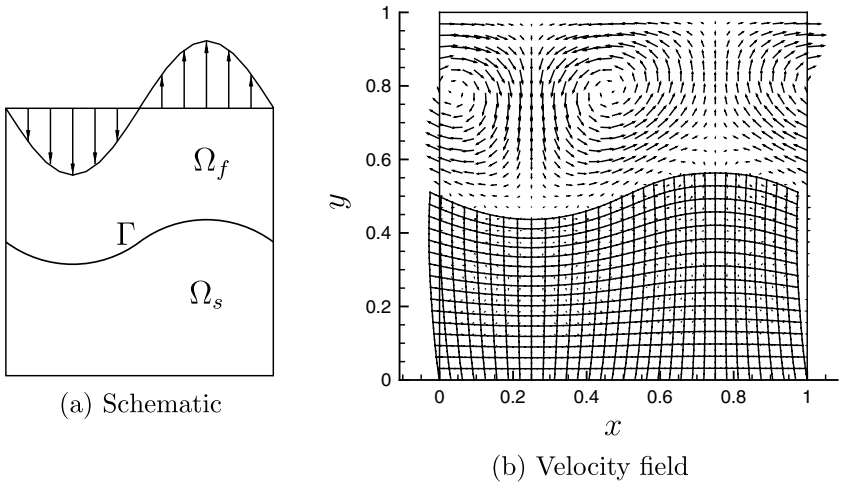


Fig. 10. The deformation of an elastic wall driven by flow at time $t = 1$.

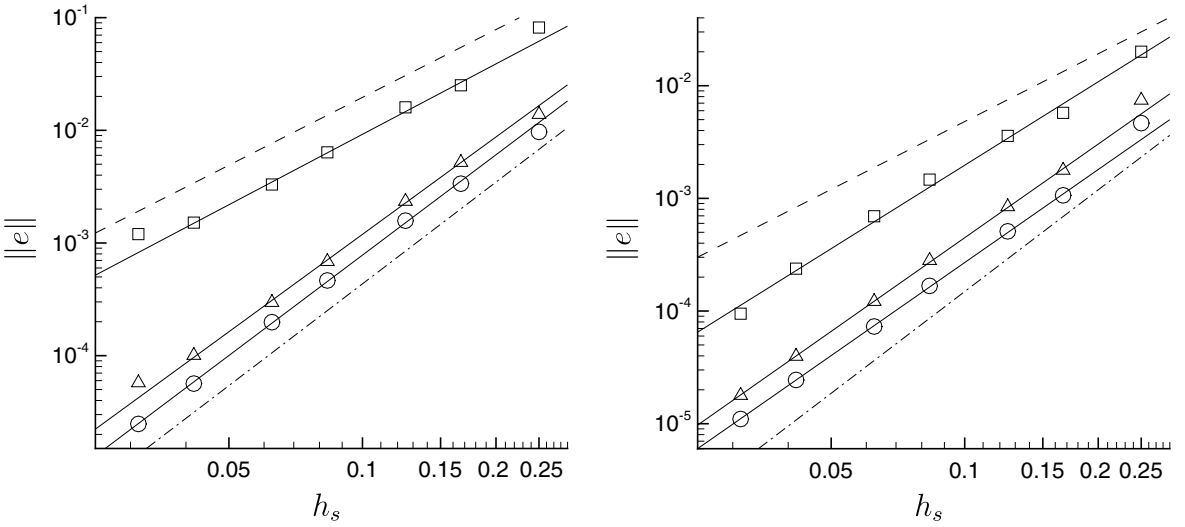
those reported by Dunne [8]. Also shown in Fig. 9 is a solution obtained via an Arbitrary Lagrangian–Eulerian method [17].

A similar configuration as shown in Fig. 10(a) is used to demonstrate convergence. The computational domain is a $[0,1] \times [0,1]$ square, the lower half of which is solid and the upper half fluid. The solid has zero moving velocity at $y = 0$, and the fluid at $y = 1$ has time-periodic impinging velocity

$$u = 0, \quad v = -(1 - \cos(2\pi t)) \sin(2\pi x). \tag{59}$$

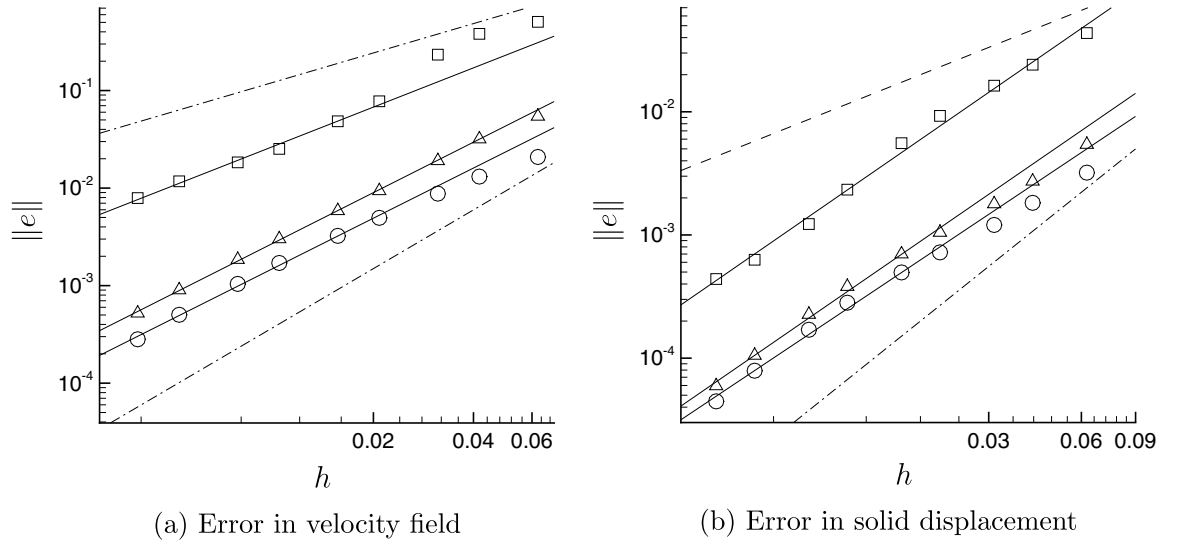
Periodic boundary conditions are specified at the left and right boundaries. The material constants are $\mu_f = 10^{-2}$ and $\mu_s = 0.25$. A visualization of the velocity field and wall deformation at $t = 1.0$ is shown in Fig. 10(b).

The undeformed wall is discretized by uniform square elements. The time step is 1/2048. The calculated velocity and solid displacement at $t = 1.5$ are compared to an accurate solution with a 64×32 Lagrangian mesh and a 512×512 Cartesian mesh to estimate error. The convergence with the Lagrangian mesh element size is shown in Fig. 11. The error in the velocity is $O(h_s^2)$ in an L_∞ norm, and $O(h_s^3)$ in L_1 and L_2 norms; the error in the solid displacement is between $O(h_s^{2.5})$ and $O(h_s^3)$ in all three norms. The convergence with the Cartesian mesh spacing is shown in Fig. 12. The velocity converges approximately as $O(h^{1.3})$ in L_∞ norm,



(a) Error in velocity field (b) Error in solid displacement

Fig. 11. The deformation of an elastic wall by flow: convergence with the solid Lagrangian mesh element size. The underlying Cartesian mesh has 512×512 cells. Plotted is the norm of the numerical error e: ○ L_1, △ L_2, □ L_∞, --- the second-order convergence line, and –·– the third-order convergence line.



(a) Error in velocity field (b) Error in solid displacement

Fig. 12. The deformation of an elastic wall by flow: convergence with Cartesian mesh spacing. The solid Lagrangian mesh has 64×32 elements. Plotted is the norm of the numerical error e: ○ L_1, △ L_2, □ L_∞, --- the first-order convergence line, and –·– the second-order convergence line.

and $O(h^{1.7})$ in L_1 and L_2 norms, and the solid displacement converges as $O(h^{1.7})$ in all three norms. The L_∞ velocity error is about 10 times the L_1 and L_2 error norms, which is in contrast to the pure solid system in Section 4.1.1, where this ratio is less than 3.

Fig. 13 shows that the pressure along a vertical line $x = 0.25$ at time $t = 0.75$ agrees well away from the interface for all three force transfer method. The immersed-interface method, as expected, captures the pressure jump at the interface near $y = 0.44$ within one mesh spacing. In contrast, the immersed-boundary method

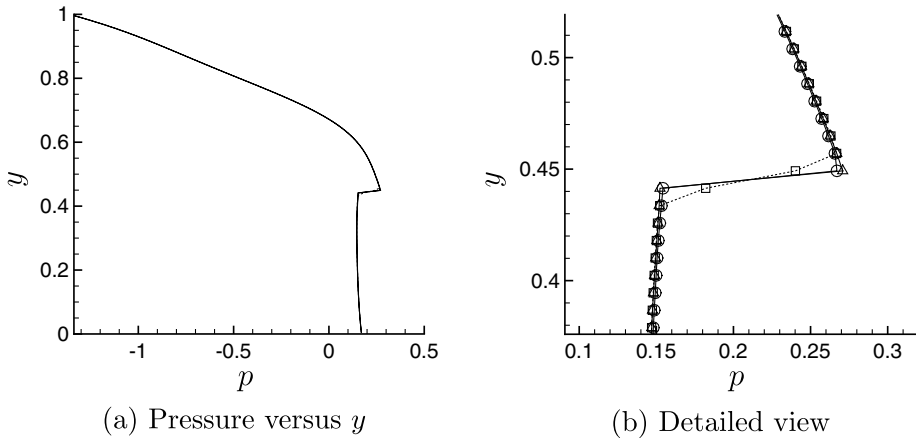


Fig. 13. Elastic wall deformed by flow: pressure on $x = 0.25$. In (a), the immersed-boundary ---, the immersed-interface —, and the force-projection –·– results are indistinguishable. In (b), □ the immersed-boundary result, △ the immersed-interface result, and ○ the force-projection result.

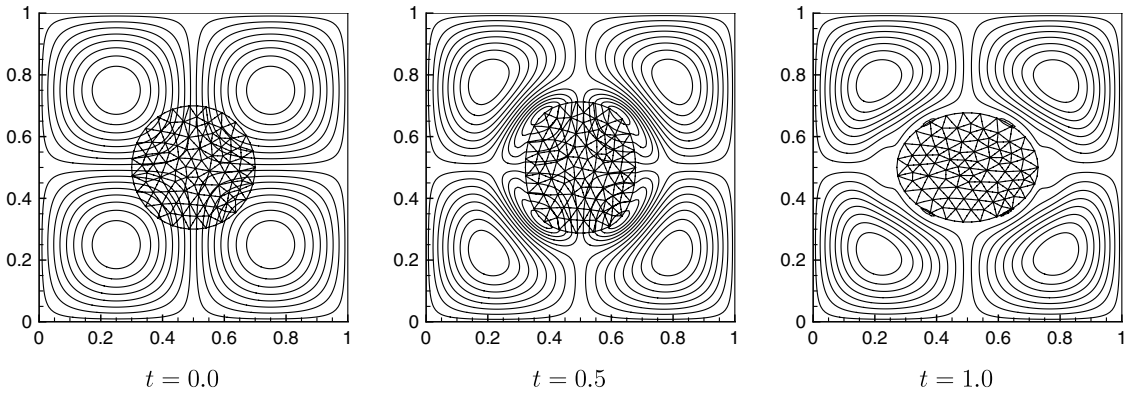


Fig. 14. Oscillating disk in fluid: streamlines and solid shape.

smears the pressure jump over three mesh spacings. The new force-projection method also appears to capture the jump within one mesh spacing.

4.1.3. An oscillating disk surrounded by fluid

This example considers a neutrally buoyant oscillating disk in a fluid. The computational domain is periodic with $0 \leqslant x \leqslant 1$ and $0 \leqslant y \leqslant 1$. The disk is initially stress free, and its undeformed shape is a circle with radius $r = 0.2$. At $t = 0$, the system is subjected to a velocity field with stream function

$$\psi = \psi_0 \sin(k_x x) \sin(k_y y), \tag{60}$$

where $\psi_0 = 5 \times 10^{-2}$ and $k_x = k_y = 2\pi$. The viscosity and the elastic constant are $\mu_f = 10^{-3}$ and $\mu_s = 1.0$.

The system evolves like a damped oscillator, with streamlines and the solid deformation as shown in Fig. 14. In these images, the discontinuity of the velocity gradient at the fluid–solid interface is indicated by the abrupt changes in the streamline direction across it. The mesh convergence for this system is similar to the elastic wall case in Section 4.1.2.

One of the attractive features of the Cartesian-mesh momentum equation solver used is its discrete energy conservations for pure fluid systems. The combined flow–structure interaction solver, however, is not expected to exactly inherit this property. We estimate its energy conservation as follows. The kinetic energy of the whole system is computed on the Cartesian mesh as

$$E_k = \frac{1}{2}\int_\Omega |\mathbf{u}(\mathbf{x})|^2 \mathrm{d}\mathbf{x} \approx \frac{h_x h_y}{2}\sum_{ij}(u^2_{i,j+\frac{1}{2}} + v^2_{i+\frac{1}{2},j}). \tag{61}$$

Likewise, the strain energy of the incompressible neo-Hookean solid is

$$E_p = \int_\Omega \frac{\mu_s}{2}(\mathrm{tr}(\mathbf{A}^T\cdot\mathbf{A}) - 2)\,\mathrm{d}\mathbf{x}, \tag{62}$$

where the deformation tensor $\mathbf{A}$ is computed elementwise through isoparametric interpolation. The total mechanical energy $E = E_k + E_p$ is dissipated by viscosity at a rate

$$\dot{E}_{\mathrm{disp}} = \int_\Omega \mu_f |\nabla\mathbf{u}|^2\,\mathrm{d}\mathbf{x}, \tag{63}$$

which is computed numerically as the sum of $|\nabla\mathbf{u}|^2$ at all Cartesian mesh points.

Analytically, the sum of the mechanical energy and the total dissipated energy,

$$E_t(t) = E_k(t) + E_p(t) + \int_0^t \dot{E}_{\mathrm{disp}}(s)\,\mathrm{d}s \tag{64}$$

should be constant. Fig. 15 shows the numerical energy balance history in one simulation, in which a 169-element Lagrangian mesh and a 128×128 Cartesian mesh are used. The computed $E_t(t)$ varies by 1.2% over these oscillations as t goes from 0 to 1. The maximum variations in E_t for all simulations in this convergence test are tabulated in Table 1. Those variations are generally consistent with solution errors, so in general, simulations with higher spatial resolutions conserve E_t better.

4.1.4. Discussion of errors due to surface force spreading

The convergence test in Section 4.1.2 shows a somewhat anomalous mesh convergence rate, which has been observed in similar systems by others [30,44]. For similar uniform mesh approaches to solve irregular domain Poisson equations using surface source singularities, it can be proved that the solution error reduces as $O(h)$ in L_∞, $O(h^2)$ in L_1 and $O(h^{1.5})$ in L_2, respectively [55]. For our flow–structure solver, the same principles apply, so we expect convergence between $O(h)$ and $O(h^2)$, as demonstrated in Section 4.1.2. An accurate prediction, however, seems intractable. One important complexity stems from the moving interface. As expected, the analysis in [55] anticipates that the error is most significant at the fluid–solid interface, where it can be magnified disproportionally since it is interpolated to update the interface location. Thus, the velocity error there will in general be propagated to the whole system in a complicated way, mostly through the transfer of surface force at the interface. The correction to the solid displacement by the solid incompressibility solver can also affect the overall accuracy, which is also hard to analyze, but expected to be small since $\mathbf{g}$ is small.

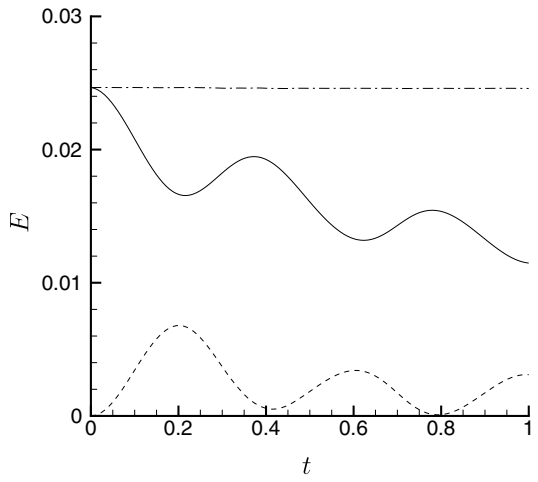


Fig. 15. Oscillating disk in fluid: energy variation. The lines show: — E_k (61), --- E_p (62), and –·– E_t (64).

Table 1
Oscillating disk in fluid: numerical energy dissipation at $t = 1.0$

N_s	h_s	$N_x(= N_y)$	e (%)
20	0.079	512	4.8
80	0.040	512	0.47
169	0.027	512	0.28
330	0.020	512	0.31
705	0.013	512	0.22
1304	0.010	512	0.15
1304	0.010	64	0.86
1304	0.010	96	0.97
1304	0.010	128	0.31
1304	0.010	192	0.15
1304	0.010	256	0.17
1304	0.010	384	0.17
1304	0.010	512	0.15

4.2. More numerical examples

4.2.1. A deformable disk in a lid-driven cavity

A lid-driven cavity is a common benchmark problem for Navier–Stokes solvers [13,26,29]. Here, our method is applied to simulate the motion of a neutrally buoyant deformable disk in a $[0,1] \times [0,1]$ lid-driven cavity. Initially, a round stress-free disk of radius 0.2 is centered at $(0.6, 0.5)$, then at $t = 0$ the top cavity wall starts moving horizontally at $U = 1$. The viscosity is $\mu_f = 10^{-2}$; two cases are simulated with $\mu_s = 0.1$ and 10, respectively.

The flow field and the solid deformation are visualized in Figs. 16 and 17. For the case where $\mu_s = 10$, the solid motion appears nearly rigid, due to the relatively weak hydrodynamic traction. However, even for this stiffer solid case, the solid disk deforms as it approaches the top lid, where its closest approach is only 0.02 as shown in Fig. 18. The disk deformation is asymmetric about the disk's vertical centerline, and lubrication forces prevent the disk from touching the lid. The disk deformation is more obvious for the $\mu_s = 0.1$ case.

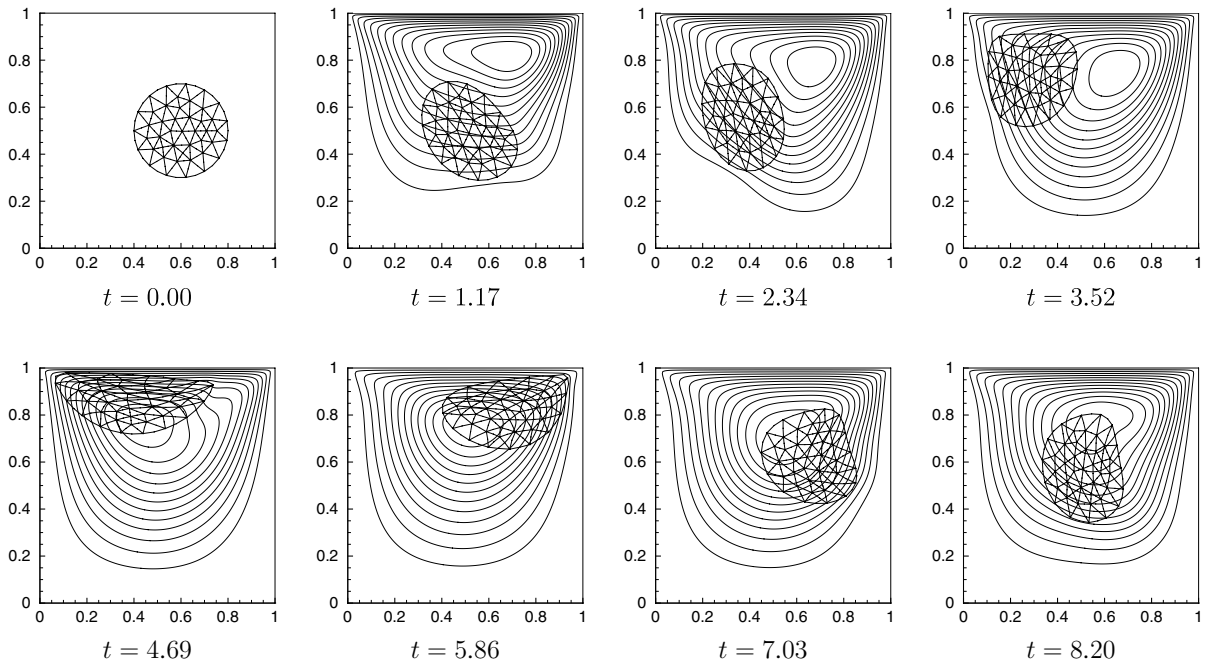


Fig. 16. Lid-driven cavity: streamlines and deformation of the disk with $\mu_s = 0.1$.

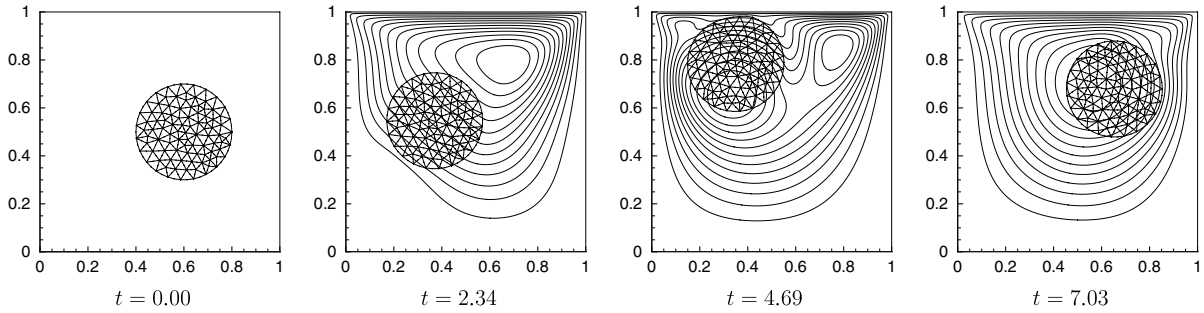


Fig. 17. Lid-driven cavity: streamlines and deformation of the disk with $\mu_s = 10$.

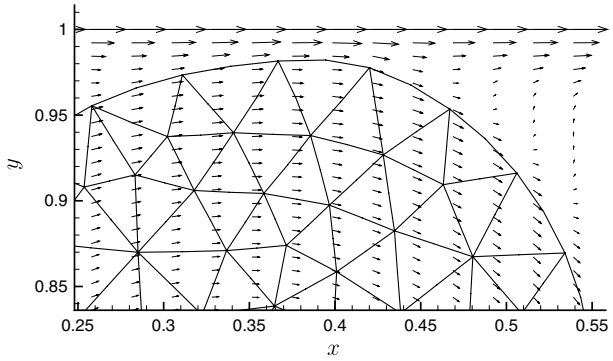


Fig. 18. Lid-driven cavity: details of the velocity field for the $\mu_s = 10$ disk.

In both cases, the solid body ends up in a fixed position near the center of the cavity, and the velocity field becomes steady.

The narrow gap between the disk and the top cavity wall must be properly resolved by the Cartesian mesh, or else the support of the distributed surface force would expand through the wall beyond the domain. For the force-projection method (36), this requires at least two Cartesian mesh cells within the gap, which is always the case for the 128×128 Cartesian mesh used.

4.2.2. A thin leaflet in an oscillating channel flow

In this example, we simulate a thin elastic leaflet in an oscillating channel flow with setup similar to that in [54]. The channel has length $L = 4$ and half-height $H = 1$ with a symmetry condition at $y = H$,

$$\frac{\partial u}{\partial y} = 0, \quad v = 0. \tag{65}$$

The oscillations are driven by Dirichlet boundary conditions on the velocity at the inflow boundary, whose side alternates according to the sign of $U_{\max}(t) = \sin(2\pi t)$,

$$u_{\text{in}}(y,t) = U_{\max}(t)\frac{y}{H}\left(2 - \frac{y}{H}\right),$$
$$v_{\text{in}}(y,t) = 0.$$

At the outflow boundary, a simple convective boundary condition [39]] is applied:

$$\frac{\partial \mathbf{u}}{\partial t} + U_{\text{adv}}\frac{\partial \mathbf{u}}{\partial x} = 0, \tag{66}$$

where U_{adv} is an advection velocity and is taken to be the average inflow velocity:

$$U_{\text{adv}}(t) = \frac{1}{H}\int_0^H u_{\text{in}}(y,t)\,\mathrm{d}y = \frac{2}{3}U_{\max}(t). \tag{67}$$

Numerically, $\mathbf{u}_{\text{out}}(y,t)$ is evolved in time by integrating (66) with the same Runge–Kutta scheme as that employed in the Navier–Stokes solver; the normal derivative $\partial\mathbf{u}/\partial x$ is computed by one-sided finite difference. The computed $\mathbf{u}_{\text{out}}(y,t)$ is supplemented by a correction velocity $\Delta\mathbf{u}$ in x-direction to balance the mass flux at the inflow and outflow boundaries. The leaflet has dimension 0.02×0.8, and its bottom is fixed at the center of the channel wall. The viscosity and the leaflet's elastic constant are $\mu_{\text{f}} = 10^{-2}$ and $\mu_{\text{s}} = 10^{2}$, respectively.

The channel is discretized by a uniform 512×128 Cartesian mesh, and the leaflet is discretized by a uniform 2×40 quadratic quadrilateral mesh. The time step is $\Delta t = 0.25h$. The flow field and the leaflet motion are visualized in Fig. 19.

4.2.3. Swimming of a two-dimensional jellyfish

A jellyfish moves itself in water by alternately contracting and relaxing a ring of muscle near its outer edge [5]. The body of the jellyfish is soft, with a density close to water, so our method is well suited to this system. In this simulation, its geometry is modeled as a two-dimensional viscoelastic body, with the density and viscosity matching that of the surrounding fluid. The top and bottom surfaces of the jellyfish are parametric curves of the form

$$\mathbf{x}(s) = (\gamma s, \beta - \alpha(\gamma s)^2 - \eta(\gamma s)^4), \quad s \in [-1,1], \tag{68}$$

with

$$\alpha = 0.03, \quad \beta = 1.0, \quad \gamma = 2.6, \quad \eta = 0.05 \tag{69}$$

for the top and

$$\alpha = 0.02, \quad \beta = 0.2006, \quad \gamma = 2.45, \quad \eta = 0.045 \tag{70}$$

for the bottom. The two curves are connected by tangent circular arcs at their ends. Fig. 20 shows the undeformed solid Lagrangian mesh.

The contracting force that drives the motion of the jellyfish is modeled as an external body force $\mathbf{b}_{\text{e}}$, which is periodic in time with period $T = 50$, and acts only near the two tips of the jellyfish body. Since in reality the contracting force is generated by the imbalance of the internal stresses within the jellyfish body, it must sum to zero. To enforce this constraint, the body force $\mathbf{b}_{\text{e}}$ is constructed such that it has the same magnitude but opposite directions at any two points symmetric about the centerline. For that, we label pairs of symmetric points as 1 and 2 as shown in Fig. 20. At the stress-free reference state, the distance between point 1 and the left tip $\mathbf{X}_{\text{tip}}$ is

$$r = |\mathbf{X}_1 - \mathbf{X}_{\text{tip}}| \tag{71}$$

and $\mathbf{b}_{\text{e}}$ is defined such that it is non-zero only for $r < 2$. In one time period, say $0 \leqslant t \leqslant T$, $\mathbf{b}_{\text{e}}$ is computed as

$$\mathbf{b}_{\text{e}}(\mathbf{x}_1,t) = \begin{cases} 10(2-r)\left(1-\frac{t}{3}\right)\frac{\mathbf{x}_{21}}{|\mathbf{x}_{21}|} & \text{for } r<2 \quad \text{and} \quad t<3, \\ 0 & \text{otherwise,} \end{cases} \tag{72}$$

$$\mathbf{b}_{\text{e}}(\mathbf{x}_2,t) = -\mathbf{b}_{\text{e}}(\mathbf{x}_1,t), \tag{73}$$

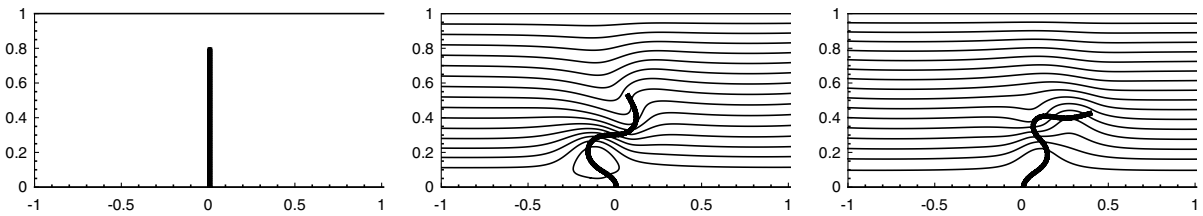


Fig. 19. Flexible leaflet in a channel: streamlines and deformation of the leaflet.

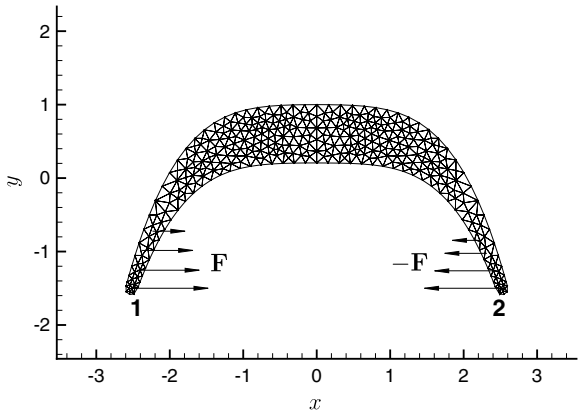


Fig. 20. A two-dimensional jellyfish.

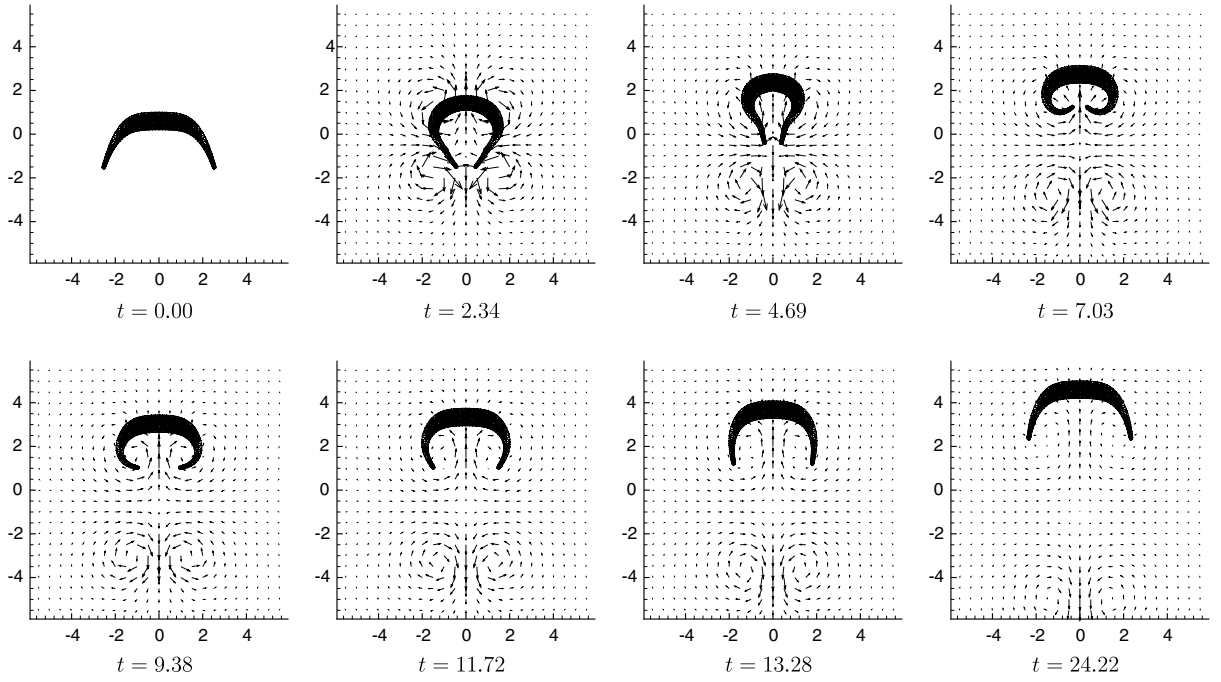


Fig. 21. Swimming of a jellyfish: velocity field and jellyfish motion.

where $\mathbf{x}_{21} = \mathbf{x}_2 - \mathbf{x}_1$ is the vector from 1 to 2 at time t.

The computational domain is periodic with $-8 \leqslant x, y \leqslant 8$ and is discretized by a 512×512 Cartesian mesh. The viscosity and the elastic constant are $\mu_f = 10^{-2}$ and $\mu_s = 5$, respectively, and the time step is $\Delta t = 1.56 \times 10^{-2}$. The velocity and the motion of the jellyfish are visualized in Fig. 21. During the contraction phase, the two tips move toward each other horizontally, producing a pair of counter-rotating vortices and pushing the jellyfish upward. The relaxation of the jellyfish body at a later time introduces another pair of vortexes that bring the surrounding fluid toward the bottom of the jellyfish body as it relaxes.

5. Conclusions and discussion

We have developed a fixed-mesh method to simulate flow–structure interaction systems with low elastic modulus. A single combined momentum equation is formulated to describe the coupled motion of the system,

which is in essence a Navier–Stokes equation supplemented by body and surface forces, which represent the flow–structure interactions. The whole velocity field is obtained by solving the momentum equation on a fixed staggered Cartesian mesh. The computational scheme is made efficient by avoiding mesh movement and regeneration and by using fast Cartesian-mesh finite-difference methods.

The transferring of solid elastic forces to the Cartesian mesh is a critical step in our method. We have considered three possible approaches: an immersed-boundary, an immersed-interface, and a new force-projection method. The immersed-boundary and force-projection methods are relatively easier to implement, and the force-projection method exactly conserves linear momentum. The immersed-interface method, on the other hand, can potentially achieve higher-order accuracy and can capture sharp jumps in pressure and velocity gradient.

By numerical tests, we show that the transfer of the elastic force from the Lagrangian mesh to the underlying Cartesian mesh is the most significant factor in reducing the numerical accuracy. In the absence of the fluid–solid interface, the calculated velocity and solid displacement are third-order accurate with respect to the Lagrangian mesh element size, and second-order accurate with respect to the Cartesian mesh spacing. When the fluid–solid interface exists, the order of accuracy decreases, with the convergence rate between second and third order with the solid Lagrangian mesh element size, and between first and second order with the Cartesian mesh spacing. The discrepancy between L_∞ and $L_{1,2}$ error norms are more significant in this case, indicating localization of errors. For fluid flow with high Reynolds number, thin boundary layers typically develop at the fluid–solid interface, so adaptive mesh refinement would help to improve the accuracy of our scheme [1,2,14,43].

The explicit Lagrangian mesh movement makes our scheme efficient, but also limits the maximum time step for stable time integration. When neglecting convection, the time step limit is inversely proportional to $\sqrt{\mu_s}$, so this explicit mesh movement is best suited for soft solid. For stiff solid, an implicit mesh movement might be preferable in order to have reasonable time step. In the immersed-boundary methods for membrane flows, implicit membrane movement has been used [35,44], where the determination of membrane positions at the next time step is formulated as a fixed-point problem and solved iteratively. Because of the similarity between our system and the membrane flow system, similar approaches should be possible.

To extend our scheme to three dimensions, the major challenge comes from the transferring of the elastic forces to the Cartesian mesh. This would be the easiest for the immersed-boundary method, which only involves pointwise interpolations and surface integrations. For the projection method, the common refinement in three dimensions is between linear tetrahedrons and the Cartesian mesh. The task of finding the vertices where straight segments intersect (Fig. 4(a)) is now replaced by the task of finding edges where planes intersect, but this is straightforward since no subsequent reconstruction is required. The algorithm will be more complicated compared to the two-dimensional case, but no fundamental difficulties are expected since all geometries involved are still straight-edged and convex. The integrand in (36) will be piecewise cubic. By applying Green's theorem twice, it can be converted to an integral of a fifth-order polynomial along the polyhedral edges, and be computed exactly by a three-point quadrature rule. Recently reported three-dimensional generalized quadrature rules [18] and re-tessellation [38] could also be used for these integrations, although they are designed for the more general problems and would be less efficient here. The immersed-interface method depends on the jump conditions, which have a very complicated form even for the simple membrane flow system [53]. Since the jump conditions involves derivatives along the interface, it demands a more sophisticated representation of the interface geometry, which introduces considerable complexity in three dimensions. The extension of the immersed-interface approach is thus expected to be most challenging.

For simplicity, we have assumed that the solid and fluid have the same density and viscosity. For mismatched densities and/or viscosities, the discontinuities can be mollified as has been done successfully for multiphase flow [3]. For example, a continuous density can be computed as

$$\rho(\mathbf{x}) = \rho_s \chi_s^h(\mathbf{x}) + \rho_f(1 - \chi_s^h(\mathbf{x})), \tag{74}$$

where χ_s^h is a mollified characteristic function for Ω_s that can be defined as

$$\chi_s^h = \chi_s * d_h, \tag{75}$$

where d_h is a smooth kernel function. The function χ_s^h changes continuously from 1 in Ω_s to 0 in Ω_f within a narrow band around the interface. Our current method can then be adopted unchanged provided a variable density and viscosity Navier–Stokes solver is used. More complex viscoelastic properties can be included by including appropriate constitutive models in (8), which can be done with little change to the overall structure of the solver.

Acknowledgments

We gratefully acknowledge the support from DOE via the Center for Simulation of Advanced Rockets at the University of Illinois at Urbana-Champaign.

References

[1] G. Agresar, J.J. Linderman, G. Tryggvason, K.G. Powell, An adaptive, Cartesian, front-tracking method for the motion, deformation and adhesion of circulating cells, J. Comput. Phys. 143 (2) (1998) 346–380.
[2] S. Bayyuk, K.G. Powell, B. van Leer, A simulation technique for 2-D unsteady inviscid flows around arbitrary moving and deforming bodies of arbitrary geometry, in: AIAA 11th Computational Fluid Dynamics Conference, Orlando, Florida, July 6–9 1993, pp. 1013–1024.
[3] Y.C. Chang, T.Y. Hou, B. Merriman, S. Osher, A level set formulation of Eulerian interface capturing methods for incompressible fluid flows, J. Comput. Phys. 124 (2) (1996) 449–464.
[4] J. Cummings, M. Aivazis, R. Samtaney, R. Radovitzky, S. Mauch, D. Meiron, A virtual test facility for the simulation of dynamic response in materials, J. Supercomput. 23 (2002) 39–59.
[5] J.O. Dabiri, S.P. Colon, J.H. Costello, M. Charib, Flow patterns generated by oblate medusan jellyfish: field measurements and laboratory analysis, J. Exp. Biol. 208 (2005) 1257–1265.
[6] G. Dasgupta, Integration within polygonal finite elements, J. Aerospace Eng. 9 (2003).
[7] R. Deiterding, R. Radovitzky, S.P. Mauch, L. Noels, J.C. Cummings, D.I. Meiron, A virtual test facility for the efficient simulation of solid material response under strong shock and detonation wave loading, Eng. Comput. 22 (2006) 325–347.
[8] T. Dunne, An Eulerian approach to fluid–structure interaction and goal-oriented mesh adaptation, Int. J. Numer. Methods Fluids 51 (2006) 1017–1039.
[9] L.J. Fauci, C.S. Peskin, A computational model of aquatic animal locomotion, J. Comput. Phys. 77 (1) (1988) 85–108.
[10] J.B. Freund, Leukocyte margination in a model microvessel, Phys. Fluids 19 (2007) 023301.
[11] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, A distributed Lagrange multiplier/fictitious domain method for particulate flows, Int. J. Multiphase Flow 25 (1999) 755–794.
[12] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, J. Périaux, A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow, J. Comput. Phys. 169 (2001) 363–426.
[13] K. Goda, A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows, J. Comput. Phys. 30 (1979) 76–95.
[14] J.A. Greenough, V. Beckner, R.B. Pember, W.Y. Crutchfield, J.B. Bell, P. Colella, An adaptive multifluid interface-capturing method for compressible flow in complex geometries, in: AIAA 26th Computational Fluid Dynamics Conference, San Diego, California, 1995.
[15] W. Hackbusch, Coupled problems in microsystem technology, in: W. Hackbusch (Ed.), Numerical Treatment of Coupled Systems, Proceedings of the 11th GAMM-Seminar Kiel, January 20–22, 1995, Notes on Numerical Fluid Mechanics, Vieweg, Wiesbaden Braunschweig, 1995.
[16] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (12) (1965) 2182–2189.
[17] C.W. Hirt, A.A. Amsden, J.L. Cook, An arbitrary Lagrangian-Eulerian computing method for all flow speeds, J. Comput. Phys. 14 (3) (1974) 227–253.
[18] D.J. Holdych, D. Noble, R.B. Secor, Quadrature rules for triangular and tetrahedral elements with generalized functions, Int. J. Numer. Methods Eng., in press, doi:10.1002/nme.2123.
[19] H.H. Hu, D.D. Joseph, M.J. Crochet, Direct simulation of fluid particle motions, Theor. Comp. Fluid Dyn. 3 (5) (1992) 285–306.
[20] H.H. Hu, N.A. Patankar, M.Y. Zhu, Direct numerical simulations of fluid–solid systems using the arbitrary Lagrangian–Eulerian technique, J. Comput. Phys. 169 (2) (2001) 427–462.
[21] T.J. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Prentice-Hall, Englewood Cliffs, 1987.
[22] R.K. Jaiman, X. Jiao, P.H. Geubelle, E. Loth, Assessment of conservative load transfer for fluid–solid interface with non-matching meshes, Int. J. Numer. Methods Eng. 64 (2005) 2014–2038.
[23] R.K. Jaiman, X. Jiao, P.H. Geubellec, E. Loth, Conservative load transfer along curved fluid–solid interface with non-matching meshes, J. Comput. Phys. 218 (1) (2006) 372–397.
[24] X. Jiao, M.T. Heath, Common-refinement based data transfer between nonmatching meshes in multiphysics simulations, Int. J. Numer. Methods Eng. 14 (6) (2004) 379–402.

[25] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, J. Comput. Phys. 171 (1) (2001) 132–150.
[26] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (1985) 308–323.
[27] M. Lai, C.S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, J. Comput. Phys. 160 (2) (2000) 705–719.
[28] L. Lee, Immersed Interface Methods for Incompressible Flow with Moving Interfaces, Ph.D. Thesis, Department of Applied Mathematics, University of Washington, 2002.
[29] L. Lee, R.J. LeVeque, An immersed interface method for incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 25 (3) (2003) 832–856.
[30] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (4) (1994) 1019–1044.
[31] R.J. LeVeque, Z. Li, Immersed interface method for Stokes flow with elastic boundaries or surface tension, SIAM J. Numer. Anal. 18 (3) (1997) 709–735.
[32] Z. Li, Immersed interface methods for moving interface problems, Numer. Algorithms 14 (1997) 269–293.
[33] W.K. Liu, Y. Liu, D. Farrell, L. Zhang, X.S. Wang, Y. Fukui, N. Patankar, Y. Zhang, C. Bajaj, Immersed finite element method and its applications to biological systems, Comput. Methods Appl. Mech. Eng. 195 (2006) 1722–1749.
[34] F. Losasso, F. Gibou, R. Fedkiw, Simulating water and smoke with an octree data structure, ACM TOG 23 (2004) 457–462.
[35] A.A. Mayao, C.S. Peskin, An implicit numerical method for fluid dynamics problems with immersed elastic boundaries, Contemp. Math. 141 (1993) 261.
[36] P. McCorquodale, P. Colella, H.A. Johansen, Cartesian grid embedded boundary method for the heat equation on irregular domains, J. Comput. Phys. 173 (2001) 620–635.
[37] R.C. McOwen, Partial Differential Equations: Methods and Equations, Prentice Hall, 1996.
[38] C.H. Min, F. Gibou, Geometric integration over irregular domains with application to level set methods, J. Comput. Phys. 226 (2) (2007) 1432–1443.
[39] M.A. Olśhanskii, V.M. Staroverov, On simulation of outflow boundary conditions in finite difference calculations for incompressible fluid, Int. J. Numer. Methods Fluids 33 (2000) 499–534.
[40] C.S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (2) (1972) 252–271.
[41] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (3) (1977) 220–252.
[42] C. Pozrikidis, Boundary Integral and Singularity Methods for Linearized Viscous Flow, Cambridge University Press, Cambridge, 1992.
[43] A.M. Roma, A Multilevel Self-adaptive Version of the Immersed Boundary Method, Ph.D. Thesis, Courant Institute of Mathematical Sciences, New York University, 1996.
[44] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (2) (1999) 509–534.
[45] P.R. Spalart, R.D. Moser, M.M. Rogers, Spectral methods for the Navier–Stokes equations with one infinite and two periodic directions, J. Comput. Phys. 96 (2) (1991) 297–324.
[46] D. Sulsky, J.U. Brackbill, A numerical method for suspension flow, J. Comput. Phys. 96 (2) (1991) 339–368.
[47] D. Sulsky, Z. Chen, H.L. Schreyer, A particle method for history-dependent materials, Comput. Methods Appl. Mech. Eng. 118 (1994) 179–196.
[48] K. Taira, T. Colonius, The immersed boundary method: a projection approach, J. Comput. Phys. 225 (2) (2007) 2118–2137.
[49] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface Cartesian grid method for simulating flows with complex moving boundaries, J. Comput. Phys. 174 (1) (2001) 380.
[50] H.S. Udaykumar, L. Tran, D.M. Belk, K.J. Vanden, An Eulerian method for computation of multimaterial impact with ENO shock-capturing and sharp interfaces, J. Comput. Phys. 186 (1) (2003) 136–177.
[51] R.W.C.P. Verstappen, A.E.P. Veldman, Symmetry-preserving discretization of turbulent flow, J. Comput. Phys. 187 (1) (2003) 343–368.
[52] X. Wang, W.K. Liu, Extended immersed boundary method using FEM and RKPM, Comput. Methods Appl. Mech. Eng. 193 (2004) 1305–1321.
[53] S. Xu, Z.J. Wang, Systematic derivation of jump conditions for the immersed interface method in three-dimensional flow simulation, SIAM J. Sci. Comput. 27 (6) (2006) 1948–1980.
[54] Z. Yu, A DLM/FD method for fluid/flexible-body interactions, J. Comput. Phys. 207 (2005) 1–27.
[55] H. Zhao, A Fixed-mesh Flow–Structure Solver for Biological Systems with Large Solid Deformations, Ph.D. Thesis, Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, 2006.
[56] X. Zhong, A new higher-order immersed interface method for multi-phase flow simulation, AIAA Paper, 2006-1294, 2006.
[57] O.C. Zienkiewicz, J.Z. Zhu, The superconvergent patch recovery and a posteriori error estimates Part 1: The recovery technique, Int. J. Numer. Methods Eng. 333 (7) (1992) 1331–1364.