Journal of Computational Physics 344 (2017) 683–723

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part II

Robert Saye

Mathematics Group, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

ARTICLE INFO

Article history:
Received 18 September 2016
Received in revised form 12 March 2017
Accepted 28 April 2017
Available online 11 May 2017

Keywords:
Interface dynamics
Discontinuous Galerkin methods
Implicitly defined meshes
Interfacial gauge methods
Level set methods
High-order

ABSTRACT

In this two-part paper, a high-order accurate implicit mesh discontinuous Galerkin (dG) framework is developed for fluid interface dynamics, facilitating precise computation of interfacial fluid flow in evolving geometries. The framework uses implicitly defined meshes—wherein a reference quadtree or octree grid is combined with an implicit representation of evolving interfaces and moving domain boundaries—and allows physically prescribed interfacial jump conditions to be imposed or captured with high-order accuracy. Part one discusses the design of the framework, including: (i) high-order quadrature for implicitly defined elements and faces; (ii) high-order accurate discretisation of scalar and vector-valued elliptic partial differential equations with interfacial jumps in ellipticity coefficient, leading to optimal-order accuracy in the maximum norm and discrete linear systems that are symmetric positive (semi)definite; (iii) the design of incompressible fluid flow projection operators, which except for the influence of small penalty parameters, are discretely idempotent; and (iv) the design of geometric multigrid methods for elliptic interface problems on implicitly defined meshes and their use as preconditioners for the conjugate gradient method. Also discussed is a variety of aspects relating to moving interfaces, including: (v) dG discretisations of the level set method on implicitly defined meshes; (vi) transferring state between evolving implicit meshes; (vii) preserving mesh topology to accurately compute temporal derivatives; (viii) high-order accurate reinitialisation of level set functions; and (ix) the integration of adaptive mesh refinement.

In part two, several applications of the implicit mesh dG framework in two and three dimensions are presented, including examples of single phase flow in nontrivial geometry, surface tension-driven two phase flow with phase-dependent fluid density and viscosity, rigid body fluid–structure interaction, and free surface flow. A class of techniques known as interfacial gauge methods is adopted to solve the corresponding incompressible Navier–Stokes equations, which, compared to archetypical projection methods, have a weaker coupling between fluid velocity, pressure, and interface position, and allow high-order accurate numerical methods to be developed more easily. Convergence analyses conducted throughout the work demonstrate high-order accuracy in the maximum norm for all of the applications considered; for example, fourth-order spatial accuracy in fluid velocity, pressure, and interface location is demonstrated for surface tension-driven two phase flow in 2D and 3D. Specific application examples include: vortex shedding in nontrivial geometry, capillary wave dynamics revealing fine-scale flow features, falling rigid bodies tumbling in unsteady flow, and free surface flow over a submersed obstacle, as well as

E-mail address: rsaye@lbl.gov.

http://dx.doi.org/10.1016/j.jcp.2017.05.003

high Reynolds number soap bubble oscillation dynamics and vortex shedding induced by a type of Plateau–Rayleigh instability in water ripple free surface flow. These last two examples compare numerical results with experimental data and serve as an additional means of validation; they also reveal physical phenomena not visible in the experiments, highlight how small-scale interfacial features develop and affect macroscopic dynamics, and demonstrate the wide range of spatial scales often at play in interfacial fluid flow.

1. Introduction to Part II

In this two-part paper, a high-order accurate implicit mesh discontinuous Galerkin (dG) framework for fluid interface dynamics is developed, with the goal of enabling precise computation of fluid flow in complex geometry, e.g., to examine how small-scale interfacial features develop and affect macroscopic dynamics. The framework uses *implicitly defined meshes*—wherein a background reference quadtree/octree grid is combined with an implicit description of fluid interfaces and boundaries—and allows physically prescribed interfacial jump conditions to be imposed or captured with high-order accuracy. Several applications of the implicit mesh dG framework in two and three dimensions are presented, including examples of single phase flow in nontrivial geometry, surface tension-driven two phase flow with phase-dependent fluid density and viscosity, rigid body fluid–structure interaction, and free surface flow.

In part one of this work—see [1]—the implicit mesh dG framework is presented in detail. Here, the framework is applied to a wide array of fluid dynamics problems. As part two concentrates mainly on the governing equations of motion, time stepping methods, and experimental validation, it may be read independently of part one. However, it is recommended that part one be read first as it provides a more thorough introduction to this work; it also contains specific implementation details that are on occasion important to the discussion in part two.

1.1. Summary of Part I

The essential idea behind the implicitly defined meshes used in this work is to combine: (i) a background reference grid, whose geometric description is relatively simple (such as the quadtrees or octrees used herein); and (ii) an implicit description of any embedded interfaces and/or the domain boundary, in such a way that the interfaces/boundaries "cut" through the background grid to subsequently define a new mesh. To avoid arbitrarily small cells, as further motivated and detailed in part one [1], the main strategy adopted in this work is to use a cell merging technique, illustrated in Fig. 1. The resulting implicitly defined mesh allows physically prescribed interfacial jump conditions to be imposed or captured with high-order accuracy. In addition, the implicit representation of moving interfaces and boundaries provides a variety of other benefits, such as: interface motion can be naturally coupled to the underlying physics; mathematical formulations, numerical discretisations, and most geometric constructions can be cast independent of the spatial dimension; and, of particular interest in this work, interface dynamics, fluid dynamics, quadrature, etc., can all be made arbitrarily high-order accurate according to a user-chosen parameter determining the order. Part one [1] details the design of a discontinuous Galerkin (dG) framework based on implicitly defined meshes—a brief outline of the relevant section is as follows (using the same section numbering):

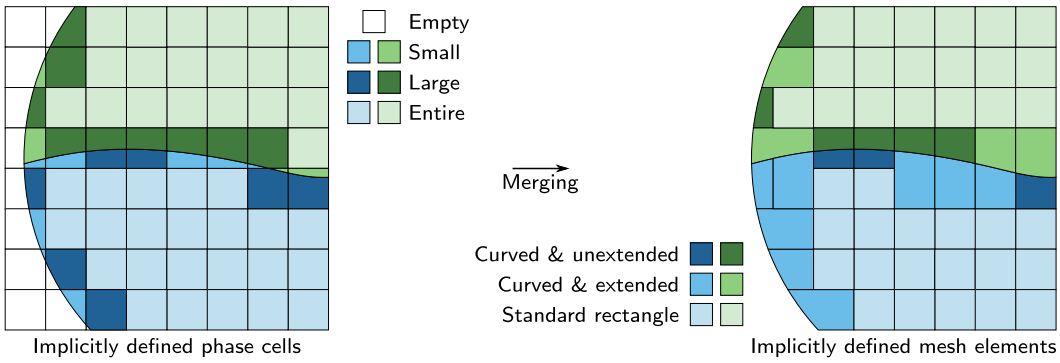

Fig. 1. Defining the mesh in the implicit mesh dG framework. (*left*) Phase cells, defined by the intersection of each phase (blue and green) with the cells of a background Cartesian/quadtree grid, are classified according to whether they fall entirely within one phase ("entire," light blue/green and rectangular) or entirely outside the domain ("empty," white), or else are denoted "partial." Partial cells are classified according to whether they have a small volume fraction (medium shade blue/green) or large intersection (dark blue/green). (*right*) Small cells are merged with neighbouring cells in the same phase to form a finite element mesh composed of standard rectangular elements and elements with curved, implicitly defined boundaries. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2.1 *Implicitly defined meshes* – The concept of implicitly defined meshes is introduced, including the definition of *phase cells* and a description of the cell merging procedure according to a user-defined small cell threshold.
2.2 *Preliminaries* – Notation, including *intraphase*, *interphase* and *boundary* faces; definition of the piecewise polynomial vector spaces V_h and V_h^d for implicitly defined meshes and for tensor-product polynomials of degree p; quadrature algorithms for computing integrals on curved elements, curved faces, or flat faces cut by the interface or domain boundary; choice of basis; computation of mass matrices; and a detailed algorithm for constructing an implicitly defined mesh.
2.3 *Local discontinuous Galerkin methods for elliptic interface problems* – Methodologies based on the local discontinuous Galerkin (LDG) [2] method to discretise a variety of elliptic interface problems, including: numerical fluxes accounting for imposed interfacial jump conditions; remarks on numerical implementation; extension to variable coefficient and vector-valued elliptic interface problems, as well as convergence tests demonstrating optimal order accuracy in the maximum norm; and the design of projection operators for incompressible fluid flow.
2.4 *Geometric multigrid* – Design of geometric multigrid algorithms to use as preconditioners for the conjugate gradient method, including: choices for the mesh hierarchy; interpolation and restriction operators; relaxation/smoothing algorithms; and results on the observed optimal-complexity of the methods when solving the elliptic interface problems discussed in §2.3.
2.5 *Advection* – Discretisation of advection terms, including upwinding and local Lax–Friedrichs fluxes.
2.6 *Evolving meshes* – Discussion on a variety of topics concerning evolving interfaces, including: design choices for evolving the interface via the level set method [3–5] or the Voronoi implicit interface method [6,7]; high-order accurate reinitialisation of a level set function as a signed distance function; conversion of a level set function, defined on mesh elements, to a level set function defined on the cells of the background quadtree/octree; transferring state (such as the velocity field) between meshes in a time stepping method; preserving mesh topology; remarks on the discrete conservation properties of the implicit mesh dG framework; and, last, some remarks on the straightforward integration of adaptive mesh refinement.

1.2. Outline for Part II

In part two, the implicit mesh dG framework is applied to a collection of fluid dynamics problems, ranging from single phase flow to multiphase flow, rigid body fluid–structure interaction, and free surface flow. Each of these problems solves, in part, the incompressible Navier–Stokes equations. For this purpose, the implicit mesh dG methodology works in tandem with a collection of techniques known as *interfacial gauge methods* [8] to compute the solution to the Navier–Stokes equations with high-order accuracy. These techniques are introduced below, and are also further motivated in part one [1].

The applications considered in this part are:

1. *Single phase flow* – implementation of a Navier–Stokes solver for a single phase incompressible fluid.
2. *Two-phase surface tension dynamics* – incompressible fluid flow driven by an interface with surface tension.
3. *Rigid body fluid–structure interaction* – motion of a rigid body embedded in a fluid-filled domain.
4. *Free surface flow* – flow of a fluid with free surface dynamics affected by surface tension and gravity.

For each of these applications, we state the equations of motion, design suitable time stepping methods, and perform several convergence analyses to verify high-order accuracy. In particular, we examine the temporal and spatial order of accuracy by using grid convergence tests, using a maximum-norm metric to confirm that high-order convergence is obtained uniformly throughout the domain. The temporal order of accuracy is examined by varying Δt, on a fixed grid with large enough resolution and high enough p to ensure spatial errors are negligible compared to temporal errors. Analogously, the spatial order of accuracy is examined by varying the element order p and typical element size h, whilst holding Δt fixed and small enough to ensure temporal errors are negligible compared to spatial errors. In addition to convergence tests, one or more example sub-applications are provided to serve as demonstrations of the framework; these include examples of bubble oscillation dynamics and Plateau–Rayleigh induced water ripples, which also serve as a form of experimental validation.

Part two then concludes with a summary of the main ideas underlying the implicit mesh dG framework and possibilities for future work. In addition, the three appendices provide further details regarding (i) axisymmetric modelling, (ii) partition of unity methods for high-order accurate smoothing, and (iii) evaluation of interface mean curvature.

2. Application to interfacial gauge methods

2.1. Single phase flow

In our first application of the implicit mesh dG framework, which also serves as an introduction to gauge methods, we consider single phase incompressible fluid flow in a fixed, static domain.

2.1.1. Equations of motion

The dynamics of an incompressible Newtonian fluid of constant density ρ and constant dynamic viscosity μ are governed by the incompressible Navier–Stokes equations

$$\left.\begin{aligned}\rho(\mathbf{u}_t+\mathbf{u}\cdot\nabla\mathbf{u}) &= -\nabla p+\mu\Delta\mathbf{u}+\mathbf{f}\\ \nabla\cdot\mathbf{u} &= 0\end{aligned}\right\}\ \text{in } \Omega, \tag{1}$$

which determine the velocity $\mathbf{u}:\Omega\to\mathbb{R}^d$ and pressure $p:\Omega\to\mathbb{R}$ of the fluid as it is subjected to an external body force $\mathbf{f}:\Omega\to\mathbb{R}^d$ in a domain $\Omega\subset\mathbb{R}^d$. We consider here boundary conditions of inflow/outflow type, wherein a boundary velocity $\mathbf{u}_\partial:\partial\Omega\to\mathbb{R}^d$ is prescribed such that $\mathbf{u}=\mathbf{u}_\partial$ on $\partial\Omega$. For example, $\mathbf{u}_\partial=0$ corresponds to a no-slip wall.

As written, the incompressible Navier–Stokes equations directly solve for the fluid velocity $\mathbf{u}$; correspondingly, a variety of numerical approaches exist for this primitive variable formulation, including the well-known and widely used projection methods pioneered by Chorin [9]. An alternative formulation of the incompressible Navier–Stokes equations, first introduced by Oseledets [10] (see also [11]), lead to a different class of methods, here referred to as gauge methods. The essential idea behind gauge methods is to replace the Navier–Stokes equations with a new set of equations determining the evolution of an auxiliary vector field $\mathbf{m}:\Omega\to\mathbb{R}^d$ and scalar field $\varphi:\Omega\to\mathbb{R}$ such that

$$\left.\begin{aligned}\rho(\mathbf{m}_t+\mathbf{u}\cdot\nabla\mathbf{u}) &= \mu\Delta\mathbf{m}+\mathbf{f}\\ \mathbf{u} &= \mathbf{m}-\nabla\varphi\\ \Delta\varphi &= \nabla\cdot\mathbf{m}\end{aligned}\right\}\ \text{in } \Omega, \tag{2}$$

subject to appropriate boundary conditions on $\partial\Omega$. Thus, instead of directly solving for the fluid velocity $\mathbf{u}$ in (1), one instead solves for $\mathbf{m}$. At every instant in time, $\mathbf{u}$ is recovered from $\mathbf{m}$ via the last two equations of (2): subject to suitable boundary conditions on the Poisson problem for φ, these equations define a *projection operator* $\mathbf{u}=\mathbb{P}(\mathbf{m})$ whose output is the divergence-free component of $\mathbf{m}$. Straightforward algebraic manipulation reveals that if $\mathbf{m}$ solves (2), then $\mathbf{u}=\mathbb{P}(\mathbf{m})$ solves the equations

$$\left.\begin{aligned}\rho(\mathbf{u}_t+\mathbf{u}\cdot\nabla\mathbf{u}) &= -\nabla(\rho\varphi_t-\mu\Delta\varphi)+\mu\Delta\mathbf{u}+\mathbf{f}\\ \nabla\cdot\mathbf{u} &= 0\end{aligned}\right\}\ \text{in } \Omega.$$

It follows that, under the assumption that solutions to the Navier–Stokes equations are unique, if $\mathbf{m}$ solves (2), then $\mathbf{u}=\mathbb{P}(\mathbf{m})$ solves (1) with pressure identified (up to an arbitrary constant) as $p=\rho\varphi_t-\mu\Delta\varphi$. As further discussed in [8] (see also [12–14]), a key point in this construction—and gauge methods in general—is that $\mathbf{m}$ is freed from having to satisfy a global divergence constraint in its evolution equation (2). This holds a variety of advantages for designing accurate and flexible time-stepping methods for incompressible fluid flow.

To complete the specification of the gauge method, we must choose suitable boundary conditions for both $\mathbf{m}$ and φ in (2), in such a way that $\mathbf{u}=\mathbb{P}(\mathbf{m})$ satisfies the correct, physically prescribed boundary conditions. This relates to the specification of a suitable projection operator, for which there are at least two guiding principles (see [8]): (i) $\mathbb{P}$ should be idempotent, and (ii) $\mathbb{P}(\cdot)\cdot\mathbf{n}$ should yield the correct normal component of the prescribed physical boundary condition, $\mathbf{u}_\partial\cdot\mathbf{n}$, independent of the input satisfying any sort of compatibility condition. These objectives can be achieved by controlling the Neumann boundary condition in the projection operator's associated Poisson problem. However, at present, we are only stating the equations of motion in the continuum, for which it suffices to choose homogeneous Neumann boundary conditions for φ. The system in (2) is then supplemented by the following boundary conditions:

$$\left.\begin{aligned}\mathbf{m}\cdot\mathbf{n} &= \mathbf{u}_\partial\cdot\mathbf{n}\\ \mathbf{m}\cdot\boldsymbol{\tau} &= \mathbf{u}_\partial\cdot\boldsymbol{\tau}+\nabla\varphi\cdot\boldsymbol{\tau}\\ \partial_n\varphi &= 0\end{aligned}\right\}\ \text{on } \partial\Omega, \tag{3}$$

where $\boldsymbol{\tau}$ is any vector tangent to $\partial\Omega$. Thus, the normal component of $\mathbf{m}$ is copied directly from $\mathbf{u}_\partial\cdot\mathbf{n}$, whereas the tangential component is coupled to $\nabla\varphi$. At first glance, in considering the numerical solution of (2)–(3), this coupling leads to a predicament because the boundary conditions on $\mathbf{m}$ depend on the projection of $\mathbf{m}$. A straightforward approach for treating this coupling is to use a time-stepping scheme that predicts boundary data for $\mathbf{m}$ by extrapolating $\nabla\varphi$ from previous time steps, similar to a multistep method. We discuss this next.

2.1.2. Temporal discretisation

For single phase flow in a static domain, a variety of high-order time stepping schemes for the incompressible Navier–Stokes equations have been developed, including multistep methods [15,16] and spectral deferred corrections [12–14,17]. In later applications, we consider fluid flow problems with evolving interfaces in which it is particularly simple to adopt a predictor–corrector time stepping scheme to achieve second order accuracy in time. To motivate these later schemes, we describe here a predictor–corrector scheme for single phase flow.

The time stepping method is a mixture of explicit and implicit schemes: advection terms are computed explicitly, whereas viscous terms are computed semi-implicitly. In particular, the coupling that arises between the gauge variables $\mathbf{m}$ and φ (as in the domain boundary condition (3)) is straightforwardly treated with a predictor of φ computed by extrapolating φ from previous time steps. This predictor is used only as a source term for boundary conditions—at the end of each time step, a new gauge variable φ is obtained, replacing the predictor. Denote by $\mathbf{u}^n$, $\mathbf{m}^n$, etc., state quantities at time step n; predictor quantities, which are first order approximations at time step $n+1$, are denoted with a $\star$. For simplicity, the time step Δt is assumed constant. With this notation, the predictor–corrector scheme for the gauge method (2)–(3) is described in Algorithm 1. Several remarks relating to the algorithms operation and implementation are in order:

Algorithm 1 Predictor–corrector scheme for the single phase, static domain gauge method (2)–(3).

1: Initialise $\mathbf{u}^0$, $\mathbf{m}^0$, and φ^0 at time $t = 0$.
2: **for** $n = 0, 1, 2, \ldots$ **do**
3: Form a second-order predictor of the gauge variable at time step $n+1$: $\varphi^\star := 2\varphi^n - \varphi^{n-1}$.
Predictor step
4: Calculate $\mathbf{m}^\star$ such that

$$\begin{cases} \rho\left(\frac{1}{\Delta t}(\mathbf{m}^\star - \mathbf{m}^n) + \mathbf{u}^n \cdot \nabla \mathbf{u}^n\right) = \mu \Delta \mathbf{m}^\star + \mathbf{f}^n & \text{in } \Omega \\ \mathbf{m}^\star \cdot \mathbf{n} = \mathbf{u}_\partial^{n+1} \cdot \mathbf{n} \text{ and } \mathbf{m}^\star \cdot \boldsymbol{\tau} = \mathbf{u}_\partial^{n+1} \cdot \boldsymbol{\tau} + \nabla \varphi^\star \cdot \boldsymbol{\tau} & \text{on } \partial\Omega. \end{cases}$$

5: Project $\mathbf{m}^\star$ to find $\mathbf{u}^\star = \mathbf{m}^\star - \nabla\tilde{\varphi}$ where $\tilde{\varphi}$ solves

$$\begin{cases} \Delta\tilde{\varphi} = \nabla \cdot \mathbf{m}^\star & \text{in } \Omega \\ \partial_n \tilde{\varphi} = \mathbf{m}^\star \cdot \mathbf{n} - \mathbf{u}_\partial^{n+1} \cdot \mathbf{n} & \text{on } \partial\Omega. \end{cases}$$

Corrector step
6: Calculate $\mathbf{m}^{n+1}$ such that

$$\begin{cases} \rho\left(\frac{1}{\Delta t}(\mathbf{m}^{n+1} - \mathbf{m}^n) + \frac{1}{2}(\mathbf{u}^n \cdot \nabla \mathbf{u}^n + \mathbf{u}^\star \cdot \nabla \mathbf{u}^\star)\right) = \frac{\mu}{2}\Delta(\mathbf{m}^n + \mathbf{m}^{n+1}) + \mathbf{f}^{n+1/2} & \text{in } \Omega \\ \mathbf{m}^{n+1} \cdot \mathbf{n} = \mathbf{u}_\partial^{n+1} \cdot \mathbf{n} \text{ and } \mathbf{m}^\star \cdot \boldsymbol{\tau} = \mathbf{u}_\partial^{n+1} \cdot \boldsymbol{\tau} + \nabla \varphi^\star \cdot \boldsymbol{\tau} & \text{on } \partial\Omega. \end{cases}$$

7: Project $\mathbf{m}^{n+1}$ to find $\mathbf{u}^{n+1} = \mathbf{m}^{n+1} - \nabla\varphi^{n+1}$ where φ^{n+1} solves

$$\begin{cases} \Delta\varphi^{n+1} = \nabla \cdot \mathbf{m}^{n+1} & \text{in } \Omega \\ \partial_n \varphi^{n+1} = \mathbf{m}^{n+1} \cdot \mathbf{n} - \mathbf{u}_\partial^{n+1} \cdot \mathbf{n} & \text{on } \partial\Omega. \end{cases}$$

8: Optionally, compute pressure at time step $n+1$ according to: $p^{n+1} = \frac{\rho}{\Delta t}(\frac{3}{2}\varphi^{n+1} - 2\varphi^n + \frac{1}{2}\varphi^{n-1}) - \mu\nabla \cdot \mathbf{m}^{n+1}$.

- In this single phase application, the sole purpose of the predictor step is to form a second-order approximation to the advection term $\mathbf{u} \cdot \nabla\mathbf{u}$. Regarding this term, note that an explicit trapezoid rule has been employed—this is to be consistent with later applications involving moving interfaces, in which utilising the trapezoidal rule is key to achieving high-order accuracy.
- To initialise the scheme, it is often sufficient to initialise $\mathbf{m}^0 = \mathbf{u}^0$ and set $\varphi^0 = 0$. (We note that more sophisticated initialisation of gauge methods may sometimes be necessary for third or higher-order accurate time stepping schemes, see [17].) In order to form the initial $\varphi^\star$ predictor at the beginning of the first time step (see line 3), we also need $\varphi^{n=-1}$: depending on the dynamics, in some cases it suffices to initialise $\varphi^{n=-1} = 0$; in other cases one can use a first-order method as part of an iterative boot-strapping scheme to initialise φ^{-1}; see, e.g., the application §2.3 for rigid body motion.
- The spatial discretisation of Algorithm 1 adopts the methods described in part one [1]. In particular, in this single phase, static domain application, all of the discrete spatial operators are independent of the time step because the implicit mesh is static. Regarding the backward Euler and Crank–Nicolson step, these steps involve solving a linear problem of the form $(I - \alpha\Delta)u = f$. Discretely, this is a simple modification of the elliptic interface problems discussed in §2.3 of part one [1]: one obtains the matrix $M + \alpha(\sum_k G_k^\mathsf{T} M G_k + \tau_0 A_0 + \sum_{i>j} \tau_{ij} A_{ij} + \tau_D A_D)$ corresponding to the heat operator—the matrix is symmetric positive definite and the same multigrid preconditioned conjugate gradient methods can be used to solve the resulting linear system, with similar convergence properties as those discussed in §2.4 of part one [1].
- Note that the two projection steps require φ satisfy Neumann boundary conditions involving $\mathbf{m}^\star \cdot \mathbf{n}$ or $\mathbf{m}^{n+1} \cdot \mathbf{n}$. Although these boundary conditions are a priori known, and seemingly should exactly cancel $\mathbf{u}_\partial^{n+1} \cdot \mathbf{n}$ to leave homogeneous boundary conditions for φ, in practice, the dG methods considered in this work only weakly enforce the boundary conditions in the backward Euler solve (for $\mathbf{m}^\star$) or Crank–Nicolson solve (for $\mathbf{m}^{n+1}$). These considerations relate to the stable implementation of the projection operator: recall that one of the design guidelines is for $\mathbb{P}(\cdot) \cdot \mathbf{n}$ to yield the correct physically prescribed boundary condition regardless of the input. We define $\mathbb{P}$ as follows:

$$\text{Given } \mathbf{m} : \Omega \to \mathbb{R}^d,\ \mathbb{P}(\mathbf{m}) = \mathbf{m} - \nabla\psi \text{ where } \begin{cases} \Delta\psi = \nabla \cdot \mathbf{m} & \text{in } \Omega \\ \partial_n \psi = \mathbf{m} \cdot \mathbf{n} - \mathbf{u}_\partial \cdot \mathbf{n} & \text{on } \partial\Omega. \end{cases}$$

  The implementation of this operator for the implicit mesh dG method is discussed in §2.3.7 of part one [1], where it is noted that, putting aside the influence of LDG penalty parameters, the discrete projection operator is idempotent. With this operator, the projection steps in Algorithm 1 can be equivalently written as $\mathbf{u}^\star = \mathbb{P}(\mathbf{m}^\star)$ and $\mathbf{u}^{n+1} = \mathbb{P}(\mathbf{m}^{n+1})$.
- Last, although computation of fluid pressure p is unnecessary in a gauge method, one may compute it by a suitable approximation of $p = \rho\varphi_t - \mu\Delta\varphi = \rho\varphi_t - \mu\nabla \cdot \mathbf{m}$. In Algorithm 1, this is accomplished with a second-order finite difference stencil for evaluating φ_t using the current and prior two values of φ.

As discussed in [8], single-phase gauge methods are closely related to particular instances of projection methods, including the second-order method of Kim & Moin [18], as well as higher-order projection methods including consistent splitting schemes [15] and slip correction schemes [16]. In particular, the treatment of the boundary condition on $\mathbf{m}$ in Algorithm 1 is reminiscent of the slip correction idea first introduced by Kim & Moin [18], as also noted by and examined in Brown et al. [19]. Ultimately, the two methodologies differ as a gauge method solves for $\mathbf{m}$, which "drifts" away from $\mathbf{u}$, whereas

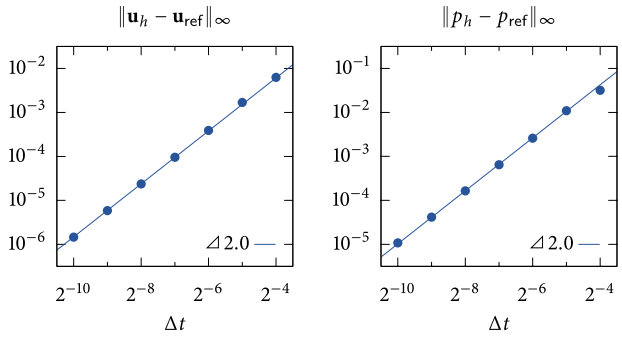

Fig. 2. Results of a temporal convergence analysis for the single phase gauge method of Algorithm 1 showing the maximum-norm error in the computed velocity field and pressure as a function of time step Δt, comparing against the exact solution (a translating Taylor–Green vortex). Data points represent measured errors at time $t = 0.5$, and the lines with indicated slope illustrate the observed order of accuracy.

projection methods solve directly for the velocity field $\mathbf{u}$ or a closely-related perturbation of $\mathbf{u}$ (denoted $\mathbf{u}^\star$ or $\tilde{\mathbf{u}}$ in the cited works). Further comments are provided in the concluding remarks of this work.

2.1.3. Convergence tests

Temporal order of accuracy. To confirm the second order accuracy of the time stepping method in Algorithm 1, we consider a two-dimensional problem with a known closed-form solution to the Navier–Stokes equations (with no external force, $\mathbf{f} = 0$), given by a translating Taylor–Green vortex. Denoting $\mathbf{u} = (u, v)$, define

$$u(x, y, t) = a + \exp(-2\lambda^2 \mu t)\sin\big(\lambda(x - at)\big)\cos\big(\lambda(y - bt)\big),$$
$$v(x, y, t) = b - \exp(-2\lambda^2 \mu t)\cos\big(\lambda(x - at)\big)\sin\big(\lambda(y - bt)\big), \text{ and}$$
$$p(x, y, t) = \tfrac{1}{4}\exp(-4\lambda^2 \mu t)\big[\cos\big(2\lambda(x - at)\big) + \cos\big(2\lambda(y - bt)\big)\big].$$

Then, $(\mathbf{u}, p)$ solves the Navier–Stokes equations with density $\rho = 1$ and viscosity μ, where (a, b) is the fixed translational velocity of the vortex and λ is an inverse length scale. We consider here a domain $\Omega = \{(x, y) : x > 0, y > 0, x^2 + y^2 < 1\}$ corresponding to the upper-right quadrant of a circle of unit radius, viscosity is set to $\mu = 0.1$, with $a = b = 1$ and $\lambda = 2$, corresponding to a Reynolds number of 20 at $t = 0$. The initial condition is given by the Taylor–Green vortex velocity field at $t = 0$, and the (time-dependent) inflow/outflow boundary conditions are given by $\mathbf{u}_\partial = \mathbf{u}|_{\partial\Omega}$. To initialise the gauge method of Algorithm 1 we set $\mathbf{m}^0 = \mathbf{u}^0$ and $\varphi^0 = \varphi^{n=-1} = 0$.

Convergence is examined by computing the maximum-norm error in the computed solution at time $t = 0.5$, when the vortex amplitude has decayed approximately 33%. Fig. 2 shows the error in velocity and pressure as a function of the repeatedly-halved time step Δt. (These results are computed using the dG framework with $p = 3$, i.e., bicubic elements, on an implicit mesh defined by a background Cartesian grid of 64×64 cells. Experiments confirmed that the resulting spatial resolution is sufficient to ensure spatial errors are negligible compared to temporal errors.) The results confirm the expected second-order accuracy of the time stepping method, for both the velocity field and pressure, in the maximum norm.

Spatial order of accuracy. To examine the spatial order of accuracy, an alternative test problem is considered. In order to observe high-order accurate spatial errors by using a second-order time stepping scheme with a time step of moderate size, it is necessary to choose a test problem with temporal errors of small magnitude. We consider here a fluid of density $\rho = 1$, viscosity $\mu = 0.1$, which is initially stationary $\mathbf{u} \equiv 0$ at $t = 0$, but subsequently forced by a time-dependent external body force $\mathbf{f}$ given by

$$\mathbf{f} = \mathbf{f}(x, y, t) = 100\frac{e^{-1/t}}{e^{-1/t} + e^{-1/(1-t)}}\big(\sin \pi x \cos \pi y, -\cos \pi x \sin \pi y\big).$$

In particular, $\mathbf{f}$ and all of its derivatives with respect to t are zero at $t = 0$. No-slip boundary conditions on $\mathbf{u}$ are imposed for all time t, and we choose a domain $\Omega = \{(x, y) : x > 0, y > 0, x^2 + y^2 < 1\}$ (in two dimensions) or $\Omega = \{(x, y, z) : x > 0, y > 0, z > 0, x^2 + y^2 + z^2 < 1\}$ (in three dimensions) corresponding to the upper-right quadrant/octant of the unit circle/sphere.

The spatial order of accuracy, which depends on p in the implicit mesh dG framework, is examined through a series of grid convergence studies, comparing against a reference solution calculated on the finest mesh (smallest h) with highest-order elements (largest p). Here, h denotes the cell size of the background uniform Cartesian grid: in 2D, the coarsest grid is 4×4 ($h = 2^{-2}$) and the finest is 128×128 ($h = 2^{-7}$); in 3D, the coarsest is $4 \times 4 \times 4$ and the finest is $32 \times 32 \times 32$. We consider $p = 2, 3, 4$, corresponding to biquadratic, bicubic, and biquartic polynomial elements in 2D, and triquadratic, tricubic, and triquartic polynomial elements in 3D. In all test cases, the time step is fixed to be $\Delta t = 10^{-4}$, empirically determined as sufficiently small so as to ensure temporal errors are negligible compared to spatial errors. Comparing against the reference solution, the maximum-norm error in fluid velocity and pressure is computed at time $t = 0.25$ for all h and p considered: the results are collected in Fig. 3 and indicate that the spatial order of accuracy is approximately at least p.

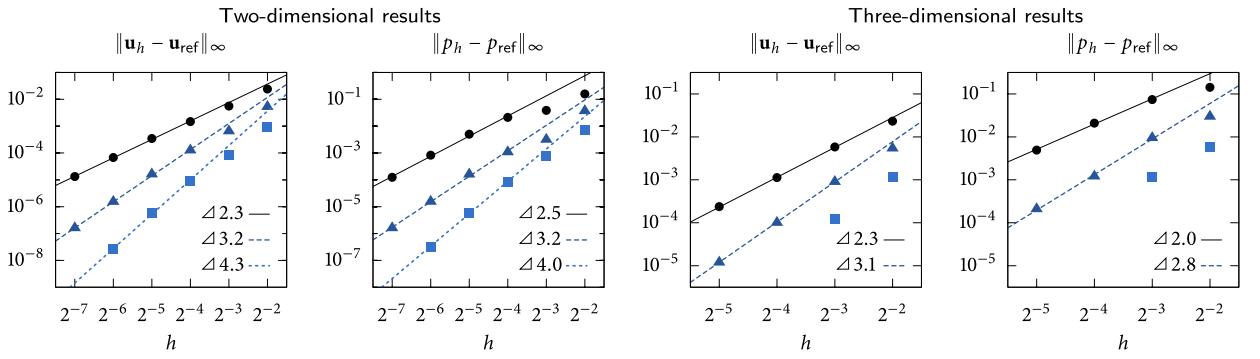

Fig. 3. Results of a spatial order of accuracy convergence analysis for the single phase gauge method of Algorithm 1 showing the maximum-norm error in the computed velocity field and pressure as a function of element size h and element order p, comparing against a reference solution computed with $h = 2^{-7}$ and $p = 4$ (in 2D, left pair), or $h = 2^{-4}$ and $p = 4$ (in 3D, right pair). Data points represent measured errors at time $t = 0.25$, whereas the lines of indicated slope illustrate the observed order of accuracy. (Lines are omitted for $p = 4$ in 3D as the results on the coarse grids are inconclusive of the asymptotic convergence rate.) • denotes $p = 2$, ▲ denotes $p = 3$, ■ denotes $p = 4$.

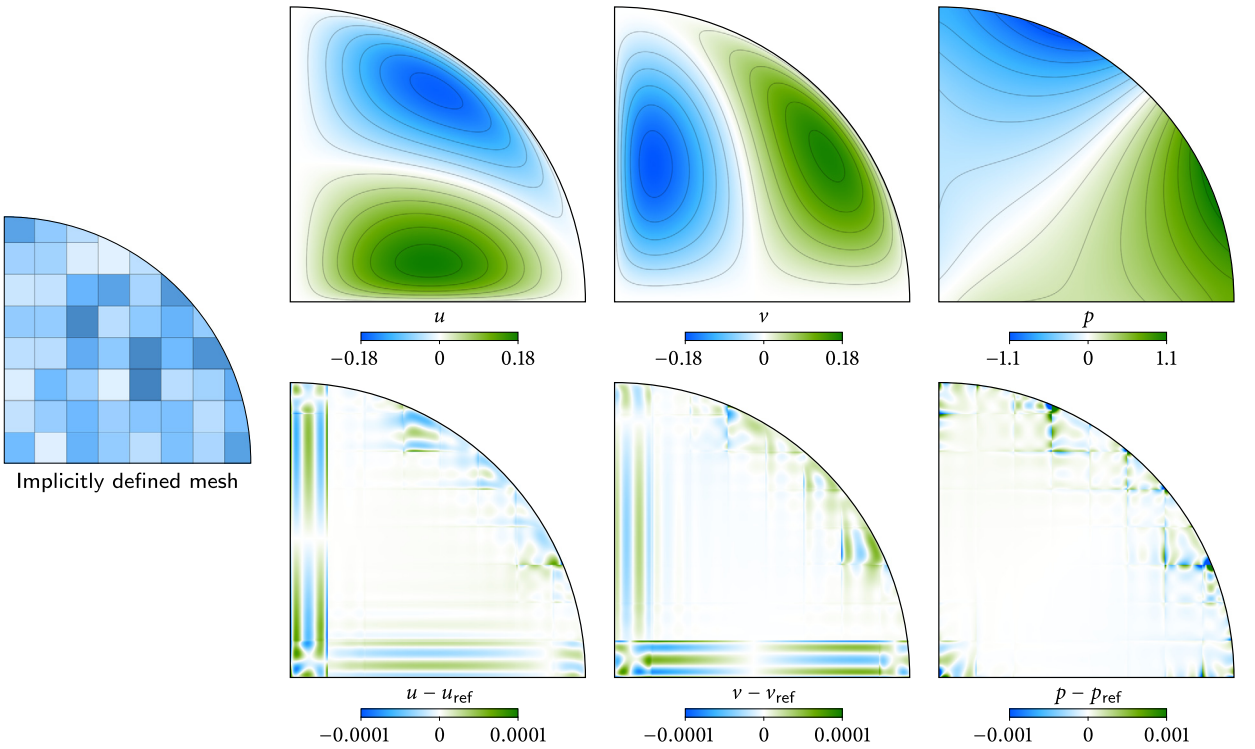

Fig. 4. Corresponding to the spatial order of accuracy grid convergence study of §2.1.3, in the specific case of an 8×8 background Cartesian grid and $p = 4$ biquartic elements, the discrete solution is shown at time $t = 0.25$. The top row illustrates the computed velocity field $\mathbf{u} = (u, v)$ and pressure field p, which are each piecewise polynomial functions on the depicted implicitly defined mesh; the bottom row plots the error in this discrete solution, which exhibits discontinuities in alignment with the element boundaries.

Note that this is suboptimal, in the sense that a finite element method using degree p polynomials has optimal order of accuracy $p + 1$. The suboptimality is attributed to the influence of the discrete projection operator $\mathbb{P}$ on unstructured meshes, see §2.3.7 of part one [1]. (In fact, the Navier–Stokes solver presented in this application has optimal order accuracy on a mesh given by a uniform Cartesian grid with periodic boundary conditions, however such results are not presented here.) Nevertheless, high-order accuracy is attained, in all quantities, in the maximum norm. Fig. 4 provides further insight into the test problem, the mesh, and the qualitative behaviour of the discrete solution: depicted is the implicitly defined mesh for the 2D test problem, based on a background 8×8 Cartesian grid, together with the computed results for $p = 4$ biquartic elements and the associated pointwise error. One can observe that the error is discontinuous across elements, as is common in dG methods, and is dominated by high frequency components; note also that the error is relatively small in magnitude despite using a coarse mesh, as one may expect for high-order accurate methods applied to smooth test problems.

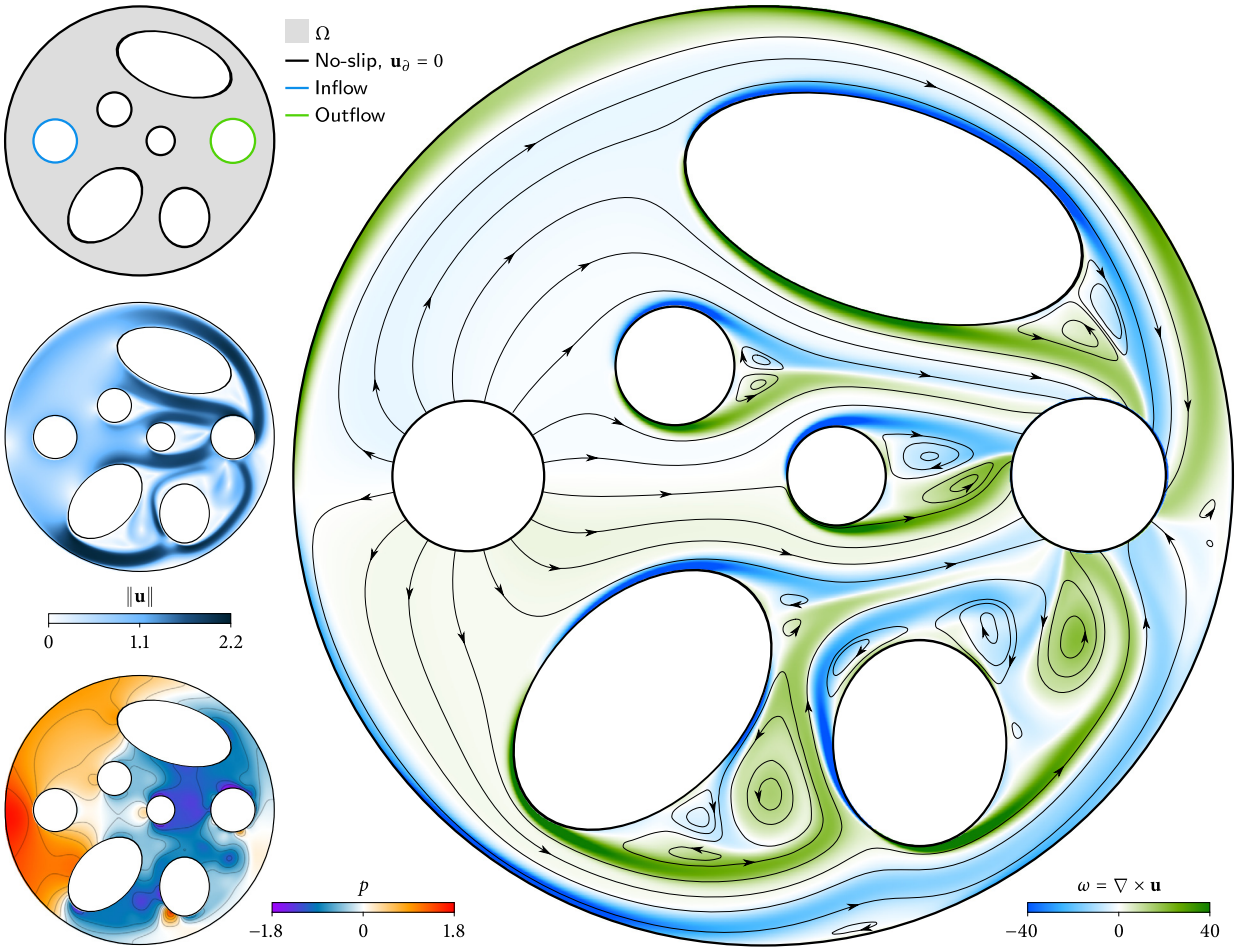

Fig. 5. Incompressible fluid flow in mildly complex geometry, computed using the single phase gauge method of Algorithm 1 and the implicit mesh dG framework with $p = 3$ bicubic elements. (*top left*) Schematic of the domain geometry, consisting of a disc with a fluid inlet (blue circle, left), outlet (green circle, right) and five ellipsoidal obstacles. (*middle left*) Speed $\|\mathbf{u}\|$ of the fluid approximately 5 time units after an initially stationary state. (*bottom left*) Pressure of the fluid, showing that pressure is greatest on the left side of the disc (owing to the impact of fluid which is forced in by the inlet) and is lowest at the centre of the eddies. (*right*) Plot of the fluid vorticity $\omega = \nabla \times \mathbf{u}$, together with instantaneous streamlines. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2.1.4. Vortex shedding in non-trivial geometry

To demonstrate the framework applied to unsteady flow in a domain with mildly complex geometry, we consider here a two-dimensional example exhibiting vortex shedding. The domain is schematically illustrated in the top-left portion of Fig. 5 and consists of a disc in which fluid flows in from an inlet on the left side (blue circle), exits from an outlet on the right side (green circle), with a number of ellipsoidal obstacles in between. The diameter of the outer disc is 3.2, the fluid has density $\rho = 1$ and viscosity $\mu = 0.001$, and the inflow/outflow boundary conditions are such that $\mathbf{u}_\partial = \pm\mathbf{n}$, where $\mathbf{n}$ is the unit normal of the inlet/outlet circles, with no-slip flow on all other boundaries. Fig. 5 illustrates the resulting dynamics approximately 5 time units after the flow has developed from an initially stationary profile, showing plots of the fluid speed, pressure, vorticity and streamlines. One can observe several eddies developed by the fluid motion as it flows from the left inlet to the right outlet. Most of the eddies are approximately stationary and exhibit a small amount of oscillation; however, results indicate the vortex structure behind the lower-right egg-shaped obstacle is less steady: indeed, vortices are shed there, owing to the greater distance between the trailing edge of the obstacle and the outflow boundary conditions which otherwise act to dampen vortex shedding. Using a velocity scale of $U = 2$ and length scale equal to the radius of the domain, the Reynolds number for this flow is approximately 3000. These results are computed with $p = 3$ bicubic polynomial elements, on an implicitly defined mesh generated by a background quadtree with a mild amount of adaptive mesh refinement: in the bulk, the mesh corresponds to a 128×128 background Cartesian grid, whereas elements near the domain boundary (i.e., within a distance of approximately 0.02) are four times smaller in order to correctly resolve the small boundary layers.

2.2. Two-phase surface tension dynamics

In the next application of the implicit mesh dG framework, we consider two-phase fluid flow driven by surface tension. These flows involve two immiscible fluids separated by a moving interface: examples include oscillating soap bubbles, ink jet dynamics, and bubble aeration. In general, the velocity field is continuous across the interface, whereas the fluid stress tensor may exhibit a jump at the interface depending on surface tension and interface curvature. To design a high-order accurate numerical scheme for computing such flows, we adopt here the interfacial gauge methods first developed in [8].

2.2.1. Equations of motion

Let Ω_1 and Ω_2 denote the time-dependent domains of the two fluids, each having density ρ_i and viscosity μ_i, and let $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$ denote the evolving interface. Adopting the same orientation for the interface unit normal $\mathbf{n}$ and jump operator $[\cdot]$ as defined in §2.2.1 of part one [1], the dynamics of surface tension-driven two-phase fluid flow is given by the incompressible Navier–Stokes equations,

$$\left.\begin{array}{c}\rho_i(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}) = -\nabla p + \mu_i\Delta\mathbf{u} \\ \nabla\cdot\mathbf{u} = 0\end{array}\right\} \text{ in } \Omega_i,$$

subject to the interfacial jump conditions

$$\left.\begin{array}{c}[\mathbf{u}] = 0 \\ [\boldsymbol{\sigma}\mathbf{n}] = -\gamma\kappa\mathbf{n}\end{array}\right\} \text{ on } \Gamma,$$

where $\boldsymbol{\sigma} = -p\mathbb{I} + \mu(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathsf{T}})$ is the fluid stress tensor, $\kappa = \nabla_s\cdot\mathbf{n}$ is the mean curvature of Γ, and γ is the coefficient of surface tension (herein assumed constant).

As in the case of single phase flow, rewriting the Navier–Stokes equations in terms of a pair of gauge variables $\mathbf{m}$ and φ leads to a wide variety of numerical advantages [8]. On the surface, there is considerable freedom in choosing how the resulting phase-dependent gauge variables are coupled across the interface. However, as discussed in [8], the most stable and accurate interfacial gauge methods arise when the associated projection operator is idempotent and satisfies $[\mathbb{P}(\cdot)]\cdot\mathbf{n} = 0$, i.e., the normal component of the projected velocity field is continuous across the interface, regardless of the input satisfying any sort of compatibility condition. Along with other considerations, this led to the design of two interfacial gauge methods: (i) one for the case in which both fluids have the same density and viscosity, and (ii) one for the case when $[\rho] \neq 0$ or $[\mu] \neq 0$. (We note that the second of these interfacial gauge methods predicts the correct dynamics even in the case that $[\rho] = 0$ and $[\mu] = 0$, however, in that special case, several simplifications in the presentation and implementation can be made; thus method (i) and (ii) are each presented.) These two interfacial gauge methods are as follows:

- *Two-phase flow with identical density and viscosity.* Assume that $\rho_1 = \rho_2$ and $\mu_1 = \mu_2$. Let $\mathbf{m}:\Omega\to\mathbb{R}^d$, $\varphi:\Omega\to\mathbb{R}$, and $q:\Omega\to\mathbb{R}$ solve

$$\left.\begin{array}{c}\rho(\mathbf{m}_t + \mathbf{u}\cdot\nabla\mathbf{u}) = -\nabla q + \mu\Delta\mathbf{m} \\ \mathbf{u} = \mathbf{m} - \nabla\varphi \\ \Delta\varphi = \nabla\cdot\mathbf{m} \\ \Delta q = 0\end{array}\right\} \text{ in } \Omega_i, \tag{4}$$

  subject to the interfacial jump conditions

$$\left.\begin{array}{c}[\varphi] = [\partial_n\varphi] = 0 \\ [\mathbf{m}] = [\partial_n\mathbf{m}] = 0 \\ [q] = \gamma\kappa,\ [\partial_n q] = 0\end{array}\right\} \text{ on } \Gamma, \tag{5}$$

  and domain boundary conditions

$$\left.\begin{array}{c}\mathbf{m}\cdot\mathbf{n} = \mathbf{u}_\partial\cdot\mathbf{n} \\ \mathbf{m}\cdot\boldsymbol{\tau} = \mathbf{u}_\partial\cdot\boldsymbol{\tau} + \nabla\varphi\cdot\boldsymbol{\tau} \\ \partial_n\varphi = \partial_n q = 0\end{array}\right\} \text{ on } \partial\Omega. \tag{6}$$

  Then, $\mathbf{u} = \mathbb{P}(\mathbf{m})$ solves the Navier–Stokes equations, with pressure identified as $p = q + \rho\varphi_t - \mu\nabla\cdot\mathbf{m}$, and the stress tensor jump condition is correctly imposed for two-phase surface tension dynamics.
- *Two-phase flow with jumps in density and viscosity.* Assume that $[\rho]$ or $[\mu]$ is not necessarily zero. Let $\mathbf{m}:\Omega\to\mathbb{R}^d$, $\varphi:\Omega\to\mathbb{R}$, and $q:\Omega\to\mathbb{R}$ solve

$$\left.\begin{array}{c}\rho_i(\mathbf{m}_t + \mathbf{u}\cdot\nabla\mathbf{u}) = -\nabla q + \nabla\cdot\left(\mu_i(\nabla\mathbf{m} + \nabla\mathbf{m}^{\mathsf{T}})\right) \\ \mathbf{u} = \mathbf{m} - \rho_i^{-1}\nabla\varphi \\ \nabla\cdot\rho_i^{-1}\nabla\varphi = \nabla\cdot\mathbf{m} \\ \Delta q = 0\end{array}\right\} \text{ in } \Omega_i, \tag{7}$$

subject to the interfacial jump conditions

$$\left.\begin{array}{c} [\varphi] = [\rho^{-1}\partial_n \varphi] = 0 \\ [\mathbf{m}\cdot\mathbf{n}] = 0,\ [\mathbf{m}\cdot\boldsymbol{\tau}] = [\rho^{-1}\nabla\varphi\cdot\boldsymbol{\tau}] \\ [\mu(\nabla\mathbf{m} + \nabla\mathbf{m}^{\mathsf{T}})\cdot\mathbf{n}] = 2[\frac{\mu}{\rho}D^2\varphi]\mathbf{n} - 2[\frac{\mu}{\rho}\Delta\varphi]\mathbf{n} \\ [q] = \gamma\kappa - [\varphi_t],\ [\partial_n q] = 0 \end{array}\right\} \text{ on } \Gamma, \tag{8}$$

and domain boundary conditions

$$\left.\begin{array}{c} \mathbf{m}\cdot\mathbf{n} = \mathbf{u}_\partial\cdot\mathbf{n} \\ \mathbf{m}\cdot\boldsymbol{\tau} = \mathbf{u}_\partial\cdot\boldsymbol{\tau} + \rho_i^{-1}\nabla\varphi\cdot\boldsymbol{\tau} \\ \partial_n\varphi = \partial_n q = 0 \end{array}\right\} \text{ on } \partial\Omega\cap\partial\Omega_i. \tag{9}$$

Then, $\mathbf{u}$ solves the Navier–Stokes equations, with pressure identified as $p = q + \varphi_t - 2\mu\nabla\cdot\mathbf{m}$, and the stress tensor jump condition is correctly imposed for two-phase surface tension dynamics.

These interfacial gauge methods are further motivated in [8]; here, a few brief remarks are appropriate:

- Note that, compared to the single phase gauge method in §2.1, in both of the above interfacial gauge methods, a term of the form $-\nabla q$ is introduced into the evolution equations for $\mathbf{m}$. The addition of such a term—even when q is an arbitrary smooth scalar—does not affect the property that the projection of $\mathbf{m}$ correctly satisfies the Navier–Stokes momentum equations. The modification only alters the formula for computing the fluid pressure p. In this case, q is used to accurately and stably account for surface tension by choosing it to be a harmonic function with interfacial jump data depending on interface curvature [8], though other constructions are possible.[1]
- The interfacial gauge method for continuous density and viscosity is particularly simple, as all of the interfacial jump conditions (except for those in q) are homogeneous. On the other hand, the second interfacial gauge method involves a coupling between the interfacial jumps of $\mathbf{m}$ and the interfacial jumps of derivatives of φ. This coupling arises because the interface stress condition $[\boldsymbol{\sigma}\mathbf{n}] = -\gamma\kappa\mathbf{n}$ involves a nontrivial interplay of the pressure and viscous stress, which cannot be simplified as in the first interfacial gauge method because neither of $[\rho]$ or $[\mu]$ is assumed zero. This coupling is in some regards similar to the coupling between $\mathbf{m}$ and $\nabla\varphi$ occurring at the domain boundary—similarly, it can be treated numerically by employing a time stepping scheme that predicts $\nabla\varphi$ and $D^2\varphi$.
- In both of the above interfacial gauge methods, we additionally solve the level set evolution equation, $\phi_t + \nabla\cdot(\mathbf{u}\phi) = 0$, to determine the evolution of the moving interface Γ.

2.2.2. Temporal discretisation

In considering the design of a time stepping method for the interfacial gauge methods (4)–(6) and (7)–(9), note that, as the interface evolves, the location of jumps change in time. Consequently, the regularity (or smoothness) of state quantities such as $\mathbf{u}$, $\mathbf{m}$, φ, etc., also change in time. Thus, it is important to use the correct discrete differential operators evaluated at the appropriate point in order to achieve spatially high-order accurate results. In the context of the implicit mesh dG framework, this relates to the fact that as the mesh evolves, so too does the discrete Laplacian, discrete gradient, and discrete divergence, etc. Among a variety of time stepping schemes taking this aspect into account, a predictor–corrector scheme is perhaps the simplest. The general operation is as follows: (i) first, a predictor interface is used to implicitly define a predictor mesh; (ii) on this mesh, the surface tension of the predictor interface is used to compute a predictor velocity field; (iii) a corrector interface is then formed; and (iv) on the corrected mesh, the surface tension of the corrected interface is used to compute second-order accurate state quantities. In this manner, it is relatively straightforward to appropriately account for the evolving mesh and its changing discrete operators.

We first consider a predictor–corrector scheme for the interfacial gauge method (4)–(6). The method was first presented schematically in [8]; here it is outlined in more detail in Algorithm 2. Several remarks are in order:

- Except for the parts of the algorithm which create a new mesh and transfer state to the new mesh, there is only one "active" mesh. All of the discrete differential operators, $\nabla\cdot$, Δ, etc., are defined relative to this mesh—this is what is meant by "adopting" the differential operators of the new mesh. In the implicit mesh dG implementation, except for the advection terms, all of the discrete differential operators in Algorithm 2 employ the LDG operators defined in §2.3 of part one [1]. For the advection terms, we use the methods in §2.5 of part one [1]; in particular, the level set update is computed with a Lax–Friedrichs flux, whereas the advection term for the velocity, $\nabla\cdot(\mathbf{u}\mathbf{u})$ is computed with an upwinding flux.

[1] Another possibility is to build q by extending the given interfacial jump condition $[q]$ away from the interface, e.g., via a closest point extension [20]. Such an extension may be cheaper to evaluate than the harmonic functions chosen here, but may compromise the stability of the time stepping scheme for the interfacial gauge method, particularly when the flow is dominated by capillary forces. See also, for example, the work of Francois et al. [21] which considers alternative methods for handling surface tension analogous to manipulating a $-\nabla q$ term in the Navier–Stokes momentum equations.

Algorithm 2 Predictor–corrector scheme for the interfacial gauge method for two-phase flow without jumps in density or viscosity, (4)–(6).

1: Define the initial implicit mesh and adopt its differential operators.
2: Initialise $\mathbf{u}^0$, $\mathbf{m}^0$, φ^0, and ϕ^0 at time $t = 0$.
3: Calculate q^0 by solving (10) using $\kappa(\phi^0)$.
4: **for** $n = 0, 1, 2, \ldots$ **do**
5: Form a second-order predictor of the gauge variable at time step $n+1$: $\varphi^\star := 2\varphi^n - \varphi^{n-1}$.
6: Define $a_\phi := \nabla \cdot (\mathbf{u}^n \phi^n)$ and $\mathbf{a_u} := \nabla \cdot (\mathbf{u}^n \mathbf{u}^n)$.
7: Define $\mathbf{f}_q := \nabla q^n$ and $\mathbf{f_m} := \mu \Delta \mathbf{m}^n$.
Predictor step
8: Update the level set function: $\phi^\star = \phi^n - \Delta t\, a_\phi$.
9: Define a predictor mesh—preserving the topology of the current mesh—using $\phi^\star$ and its implicitly defined interface $\Gamma^\star$.
10: Transfer all necessary quantities to the new mesh and adopt its differential operators.
11: Calculate $q^\star$ by solving (10) using $\kappa(\phi^\star)$.
12: Calculate $\mathbf{m}^\star$ such that

$$\begin{cases} \rho\left(\frac{1}{\Delta t}(\mathbf{m}^\star - \mathbf{m}^n) + \mathbf{a_u}\right) = -\nabla q^\star + \mu \Delta \mathbf{m}^\star & \text{in } \Omega_i \\ [\mathbf{m}^\star] = [\partial_n \mathbf{m}^\star] = 0 & \text{on } \Gamma \\ \mathbf{m}^\star \cdot \mathbf{n} = \mathbf{u}_\partial^{n+1} \cdot \mathbf{n} \text{ and } \mathbf{m}^\star \cdot \boldsymbol{\tau} = \mathbf{u}_\partial^{n+1} \cdot \boldsymbol{\tau} + \nabla \varphi^\star \cdot \boldsymbol{\tau} & \text{on } \partial\Omega. \end{cases}$$

13: Project $\mathbf{m}^\star$ to find $\mathbf{u}^\star = \mathbb{P}(\mathbf{m}^\star)$.
Corrector step
14: Define $\mathbf{a}_\mathbf{u}^\star := \nabla \cdot (\mathbf{u}^\star \mathbf{u}^\star)$.
15: Update the level set function: $\phi^{n+1} = \phi^n - \frac{1}{2}\Delta t\left(a_\phi + \nabla \cdot (\mathbf{u}^\star \phi^\star)\right)$.
16: Define the new mesh—preserving the topology of the current mesh—using ϕ^{n+1} and its implicitly defined interface Γ^{n+1}.
17: Transfer all necessary quantities to the new mesh and adopt its differential operators.
18: Calculate q^{n+1} by solving (10) using $\kappa(\phi^{n+1})$.
19: Calculate $\mathbf{m}^{n+1}$ such that

$$\begin{cases} \rho\left(\frac{1}{\Delta t}(\mathbf{m}^{n+1} - \mathbf{m}^n) + \frac{1}{2}(\mathbf{a_u} + \mathbf{a}_\mathbf{u}^\star)\right) = -\frac{1}{2}(\mathbf{f}_q + \nabla q^{n+1}) + \frac{1}{2}\left(\mathbf{f_m} + \mu \Delta \mathbf{m}^{n+1}\right) & \text{in } \Omega_i \\ [\mathbf{m}^{n+1}] = [\partial_n \mathbf{m}^{n+1}] = 0 & \text{on } \Gamma \\ \mathbf{m}^{n+1} \cdot \mathbf{n} = \mathbf{u}_\partial^{n+1} \cdot \mathbf{n} \text{ and } \mathbf{m}^{n+1} \cdot \boldsymbol{\tau} = \mathbf{u}_\partial^{n+1} \cdot \boldsymbol{\tau} + \nabla \varphi^\star \cdot \boldsymbol{\tau} & \text{on } \partial\Omega. \end{cases}$$

20: Project $\mathbf{m}^{n+1}$ to find $\mathbf{u}^{n+1} = \mathbb{P}(\mathbf{m}^{n+1})$ and $\nabla \varphi^{n+1} = \mathbf{m}^{n+1} - \mathbf{u}^{n+1}$.
21: Compute $\partial_t \varphi \approx \varphi_t^{n+1/2} := \frac{1}{\Delta t}(\varphi^{n+1} - \varphi^n)$.
22: Optionally, compute pressure at time step $n+1$ according to: $p^{n+1} = q^{n+1} + \frac{\rho}{2}(3\varphi_t^{n+1/2} - \varphi_t^{n-1/2}) - \mu \nabla \cdot \mathbf{m}^{n+1}$.
23: If the remeshing on line 9 signalled a topology update, remesh without preserving the topology, transfer all quantities to the new mesh, and adopt its differential operators.

- The projection operator $\mathbb{P}$ employed by Algorithm 2 is defined as follows:

$$\text{Given } \mathbf{m} : \Omega \to \mathbb{R}^d,\ \mathbb{P}(\mathbf{m}) = \mathbf{m} - \nabla\psi \text{ where } \begin{cases} \Delta\psi = \nabla \cdot \mathbf{m} & \text{in } \Omega \\ [\psi] = 0,\ [\partial_n \psi] = [\mathbf{m} \cdot \mathbf{n}] & \text{on } \Gamma \\ \partial_n \psi = \mathbf{m} \cdot \mathbf{n} - \mathbf{u}_\partial \cdot \mathbf{n} & \text{on } \partial\Omega. \end{cases}$$

  Its implementation is discussed in §2.3.7 of part one [1].
- To compute q, we solve the elliptic interface problem

$$\begin{cases} \Delta q = 0 & \text{in } \Omega \\ [q] = \gamma\kappa(\phi),\ [\partial_n q] = 0 & \text{on } \Gamma \\ \partial_n q = 0 & \text{on } \partial\Omega, \end{cases} \tag{10}$$

  where $\kappa(\phi)$ is the computed mean curvature of the given level set function ϕ (either ϕ^n or the predictor $\phi^\star$). To compute the mean curvature, the method outlined in Appendix C is adopted.
- Note that the corrector step uses an explicit trapezoid rule for the advection term, i.e., $\frac{1}{2}(\mathbf{a_u} + \mathbf{a}_\mathbf{u}^\star)$, equivalently, $\frac{1}{2}(\nabla^n \cdot (\mathbf{u}^n \mathbf{u}^n) + \nabla^\star \cdot (\mathbf{u}^\star \mathbf{u}^\star))$, where $\nabla^n \cdot$ and $\nabla^\star \cdot$ denote the discrete implementation of the advection operator for the mesh at time step n and the predictor mesh, respectively. This choice relates to the property that the regularity of $\mathbf{u}^n$ and $\mathbf{u}^\star$ conforms precisely to Γ^n and $\Gamma^\star$, respectively. In other words, although $\mathbf{u}$ is continuous across the interface, derivatives of $\mathbf{u}$ are generally discontinuous. It follows that one cannot use the explicit midpoint method, for example, because the regularity of the linear combination $\frac{1}{2}(\mathbf{u}^n + \mathbf{u}^\star)$ does not coincide with any well-defined interface; on the other hand, the explicit trapezoid rule can be used, as the linear combination $\nabla^n \cdot (\mathbf{u}^n \mathbf{u}^n) + \nabla^\star \cdot (\mathbf{u}^\star \mathbf{u}^\star)$ yields high-order spatial accuracy. A similar consideration applies to the curvature source term, wherein an explicit trapezoid rule is used to calculate $\frac{1}{2}(\mathbf{f}_q + \nabla q^{n+1}) = \frac{1}{2}(\nabla^n q^n + \nabla^{n+1} q^{n+1})$.
- Extensive numerical experiments indicate that this treatment of surface tension—i.e., using a harmonic q with jump conditions depending on curvature within an explicit trapezoid method—leads to a stable method: numerical methods for surface tension driven flow often entail a time-step constraint of $\Delta t < C\sqrt{(\rho_1 + \rho_2)/\gamma}\, h^{3/2}$ relating to a grid-dependent capillary wave speed, where C is an $\mathcal{O}(0.1)$ constant, ρ_1 and ρ_2 are the fluid densities on either side of the interface, and h is the smallest element size [22]. However, no such stability constraint has been observed necessary for the above predictor–corrector scheme for the interfacial gauge method (4)–(6) (nor for the interfacial gauge method (7)–(9) that

implements surface tension in a similar way). Instead, experiments indicate that the only timestep constraint required for stability is the usual advective constraint of $\Delta t \leq C \min h/|\mathbf{u}|$, where h is the local element size and $|\mathbf{u}|$ is the local fluid speed: empirically, $C \approx 0.05$ – 0.1 suffices, consistent with typical stability constraints for dG methods of the chosen order (wherein, more generally, C decreases with larger p [23]).
- Last, it is typically straightforward to initialise this interfacial gauge method by choosing $\mathbf{m}^0 = \mathbf{u}^0 = \mathbf{u}(t=0)$ and $\varphi^0 = 0$ at $t = 0$.

Before considering a predictor-scheme for the interfacial gauge method (7)–(9), we first discuss an important aspect of the remeshing procedure, common to both time stepping schemes. Referring to Algorithm 2, on lines 9 and 16, the new meshes are created by "preserving the topology" of the current mesh. This procedure refers to that described in §2.6.4 of part one [1], which is designed to create an implicit mesh using the same cell-merging procedure as the existing mesh, such that there is a one-to-one correspondence between elements on the old mesh and the new. Besides the simple state transference operator that ensues (i.e., nodal coefficients of each state vector are preserved), there is, in addition, an important reason for preserving the mesh topology, relating to the accurate calculation of $\partial_t \varphi$. By way of motivation, suppose that the topology is not preserved: then, the state transfer—due to the use of local L^2 projections whenever elements are created or destroyed—can create a high-order spatial error of magnitude $\mathcal{O}(h^{p+1})$, independent of the size of Δt. Then, if one was to estimate $\partial_t \varphi \approx \frac{1}{\Delta t}(\varphi^{n+1} - \varphi^n)$, the divided difference involves an error of the form $\mathcal{O}(h^{p+1}/\Delta t)$, which is unbounded as $\Delta t \to 0$. Thus, the act of projecting the state from one mesh to another introduces an error uncorrelated with time, preventing accurate calculation of temporal derivatives. This issue is avoided if the topology of the mesh is preserved. Doing so means the state transference operator is simple injection, which does not introduce any errors uncorrelated with time, even when interfacial elements change shape. In this setting, the divided difference $\frac{1}{\Delta t}(\varphi^{n+1} - \varphi^n)$ converges, and is a second-order accurate approximation of $\partial_t \varphi$ at time step $n + \frac{1}{2}$. Though not strictly necessary for Algorithm 2, this procedure allows one to calculate $\partial_t \varphi$ in the computation of fluid pressure $p = q + \rho \varphi_t - \mu \nabla \cdot \mathbf{m}$. Clearly, however, one cannot preserve the mesh topology indefinitely, as the interface will eventually move far enough to cross a cell in the background quadtree/octree, signalling that the cell-merging procedure must necessarily change behaviour, as discussed in §2.6.4 of part one [1]. Whenever this event is signalled, then we remesh (see line 23), allowing the topology to change, but only after computing $\varphi_t^{n+1/2}$.

We now consider a predictor–corrector scheme for the interfacial gauge method (7)–(9) for the case in which density or viscosity has jumps. The scheme is detailed in Algorithm 3 and has much in common with Algorithm 2. Several remarks are in order:

- As in the previous case, all of the discrete differential operators, $\nabla\cdot$, ∇, etc., are defined relative to the current mesh. We employ the LDG discretisation of the stress tensor divergence operator, as discussed in §2.3.5 of part one [1], whereas, for the advection terms, the level set update continues to use a Lax–Friedrichs flux, while $\nabla \cdot (\mathbf{u}\,\mathbf{u})$ is computed with an upwind flux.
- The projection operator $\mathbb{P}$ employed by Algorithm 3 is defined as follows:

$$\text{Given } \mathbf{m} : \Omega \to \mathbb{R}^d,\ \mathbb{P}(\mathbf{m}) = \mathbf{m} - \tfrac{1}{\rho}\nabla\psi, \text{ where } \begin{cases} \nabla \cdot \rho_i^{-1} \nabla \psi = \nabla \cdot \mathbf{m} & \text{in } \Omega_i \\ [\psi] = 0 \text{ and } [\rho^{-1}\partial_n \psi] = [\mathbf{m}\cdot\mathbf{n}] & \text{on } \Gamma \\ \rho_i^{-1}\partial_n \psi = \mathbf{m}\cdot\mathbf{n} - \mathbf{u}_\partial \cdot \mathbf{n} & \text{on } \partial\Omega \cap \partial\Omega_i. \end{cases}$$

  The discretisation of the projection operator is similar to that described in §2.3.7 of part one [1], requiring only minor modifications to include a diagonal matrix corresponding to the $1/\rho$ ellipticity coefficient. In particular, one can show that, putting aside the penalty parameters of the LDG method, the discrete projection operator is idempotent.
- To compute q, we solve the elliptic interface problem

$$\begin{cases} \Delta q = 0 & \text{in } \Omega \\ [q] = \gamma\kappa(\phi) - [\varphi_t],\ [\partial_n q] = 0 & \text{on } \Gamma \\ \partial_n q = 0 & \text{on } \partial\Omega, \end{cases} \tag{11}$$

  where $\kappa(\phi)$ is the computed mean curvature of the level set function ϕ (either ϕ^n or the predictor $\phi^\star$), and $[\varphi_t]$ is computed from the jump of the predictor $\varphi_t^\star$. To compute the mean curvature, the method outlined in Appendix C is again adopted.
- For this interfacial gauge method, one must compute interfacial jumps in $\frac{\mu}{\rho}D^2\varphi$ and $\frac{\mu}{\rho}\Delta\varphi$, as these determine the interfacial jump conditions in the backward Euler solve for $\mathbf{m}^\star$ and the Crank–Nicolson solve for $\mathbf{m}^{n+1}$. To calculate the Hessian of φ, we use the discrete gradient operator G_i (which approximates ∂_{x_i}) together with its adjoint, to compute $(D^2\varphi)_{ij} = \frac{1}{2}(M^{-1}G_i^{\mathsf{T}}MG_j + M^{-1}G_j^{\mathsf{T}}MG_i)\varphi$. Rather than, say, computing $G_iG_j\varphi$ (which, due to the one-sided directional principle defining the discrete gradient (see §2.3 of part one [1]), "upwinds twice" in the same direction to yield a biased one-sided stencil), this discrete operator for evaluating D^2 has a discrete stencil which is more symmetric; moreover, the trace of the resulting discrete Hessian coincides precisely with the discrete Laplacian. In highly-resolved situations, experiments indicate that this approximation to the Hessian of φ results in a stable and accurate interfacial gauge method; however, in only mildly-resolved test cases, the accuracy of the computed velocity field and pressure is more

Algorithm 3 Predictor–corrector scheme for the interfacial gauge method for two-phase flow with jumps in density or viscosity, (7)–(9).

1: Define the initial implicit mesh and adopt its differential operators.
2: Initialise $\mathbf{u}^0$, $\mathbf{m}^0$, φ^0, and ϕ^0 at time $t = 0$.
3: Determine an appropriate φ^{-1}, $\varphi_t^{-1/2}$ and $\varphi_t^{-3/2}$.
4: Calculate q^0 by solving (11) using $\kappa(\phi^0)$ and $\varphi_t = \frac{3}{2}\varphi_t^{-1/2} - \varphi_t^{-3/2}$.
5: **for** $n = 0, 1, 2, \ldots$ **do**
6: Form a second-order predictor of the gauge variable at time step $n+1$: $\varphi^\star := 2\varphi^n - \varphi^{n-1}$.
7: Form a second-order predictor of $\partial_t\varphi$ at time step $n+1$: $\varphi_t^\star := \frac{5}{2}\varphi_t^{n-1/2} - \frac{3}{2}\varphi_t^{n-3/2}$.
8: Define $a_\phi := \nabla\cdot(\mathbf{u}^n\phi^n)$ and $\mathbf{a_u} := \nabla\cdot(\mathbf{u}^n\mathbf{u}^n)$.
9: Define $\mathbf{f}_q := \nabla q^n$ and $\mathbf{f_m} := \nabla\cdot\big(\mu(\nabla\mathbf{m}^n + \nabla\mathbf{m}^{n\,\mathsf{T}})\big)$.
Predictor step
10: Update the level set function: $\phi^\star = \phi^n - \Delta t\, a_\phi$.
11: Define a predictor mesh—preserving the topology of the current mesh—using $\phi^\star$ and its implicitly defined interface $\Gamma^\star$.
12: Transfer all necessary quantities to the new mesh and adopt its differential operators.
13: Calculate $q^\star$ by solving (11) using $\kappa(\phi^\star)$ and $\varphi_t^\star$.
14: Calculate $\mathbf{m}^\star$ such that

$$\begin{cases} \rho_i\big(\frac{1}{\Delta t}(\mathbf{m}^\star - \mathbf{m}^n) + \mathbf{a_u}\big) = -\nabla q^\star + \nabla\cdot\big(\mu_i(\nabla\mathbf{m}^\star + \nabla\mathbf{m}^{\star\,\mathsf{T}})\big) & \text{in } \Omega_i \\ [\mathbf{m}^\star\cdot\mathbf{n}] = 0 \text{ and } [\mathbf{m}^\star\cdot\boldsymbol{\tau}] = [\rho^{-1}\nabla\varphi^\star\cdot\boldsymbol{\tau}] & \text{on } \Gamma \\ [\mu(\nabla\mathbf{m}^\star + \nabla\mathbf{m}^{\star\,\mathsf{T}})\cdot\mathbf{n}] = 2[\frac{\mu}{\rho}D^2\varphi^\star]\mathbf{n} - 2[\frac{\mu}{\rho}\Delta\varphi^\star]\mathbf{n} & \text{on } \Gamma \\ \mathbf{m}^\star\cdot\mathbf{n} = \mathbf{u}_\partial^{n+1}\cdot\mathbf{n} \text{ and } \mathbf{m}^\star\cdot\boldsymbol{\tau} = \mathbf{u}_\partial^{n+1}\cdot\boldsymbol{\tau} + \rho_i^{-1}\nabla\varphi^\star\cdot\boldsymbol{\tau} & \text{on } \partial\Omega\cap\partial\Omega_i. \end{cases}$$

15: Project $\mathbf{m}^\star$ to find $\mathbf{u}^\star = \mathbb{P}(\mathbf{m}^\star)$.
Corrector step
16: Define $\mathbf{a}_\mathbf{u}^\star := \nabla\cdot(\mathbf{u}^\star\mathbf{u}^\star)$.
17: Update the level set function: $\phi^{n+1} = \phi^n - \frac{1}{2}\Delta t\big(a_\phi + \nabla\cdot(\mathbf{u}^\star\phi^\star)\big)$.
18: Define the new mesh—preserving the topology of the current mesh—using ϕ^{n+1} and its implicitly defined interface Γ^{n+1}.
19: Transfer all necessary quantities to the new mesh and adopt its differential operators.
20: Calculate q^{n+1} by solving (11) using $\kappa(\phi^{n+1})$ and $\varphi_t^\star$.
21: Calculate $\mathbf{m}^{n+1}$ such that

$$\begin{cases} \rho_i\big(\frac{1}{\Delta t}(\mathbf{m}^{n+1} - \mathbf{m}^n) + \frac{1}{2}(\mathbf{a_u} + \mathbf{a}_\mathbf{u}^\star)\big) = -\frac{1}{2}(\mathbf{f}_q + \nabla q^{n+1}) + \frac{1}{2}\big[\mathbf{f_m} + \nabla\cdot\big(\mu_i(\nabla\mathbf{m}^{n+1} + \nabla\mathbf{m}^{n+1\,\mathsf{T}})\big)\big] & \text{in } \Omega_i \\ [\mathbf{m}^{n+1}\cdot\mathbf{n}] = 0 \text{ and } [\mathbf{m}^{n+1}\cdot\boldsymbol{\tau}] = [\rho^{-1}\nabla\varphi^\star\cdot\boldsymbol{\tau}] & \text{on } \Gamma \\ [\mu(\nabla\mathbf{m}^{n+1} + \nabla\mathbf{m}^{n+1\,\mathsf{T}})\cdot\mathbf{n}] = 2[\frac{\mu}{\rho}D^2\varphi^\star]\mathbf{n} - 2[\frac{\mu}{\rho}\Delta\varphi^\star]\mathbf{n} & \text{on } \Gamma \\ \mathbf{m}^{n+1}\cdot\mathbf{n} = \mathbf{u}_\partial^{n+1}\cdot\mathbf{n} \text{ and } \mathbf{m}^{n+1}\cdot\boldsymbol{\tau} = \mathbf{u}_\partial^{n+1}\cdot\boldsymbol{\tau} + \rho_i^{-1}\nabla\varphi^\star\cdot\boldsymbol{\tau} & \text{on } \partial\Omega\cap\partial\Omega_i. \end{cases}$$

22: Project $\mathbf{m}^{n+1}$ to find $\mathbf{u}^{n+1} = \mathbb{P}(\mathbf{m}^{n+1})$ and $\nabla\varphi^{n+1} = \mathbf{m}^{n+1} - \mathbf{u}^{n+1}$.
23: Compute $\varphi_t^{n+1/2} := \frac{1}{\Delta t}(\varphi^{n+1} - \varphi^n)$.
24: Optionally, compute pressure at time step $n+1$ according to: $p^{n+1} = q^{n+1} + (\frac{3}{2}\varphi_t^{n+1/2} - \frac{1}{2}\varphi_t^{n-1/2}) - 2\mu\nabla\cdot\mathbf{m}^{n+1}$.
25: If the remeshing on line 11 signalled a topology update, remesh without preserving the topology, transfer all quantities to the new mesh, and adopt its differential operators.

susceptible to noise present in the computed discrete Hessian. In under-resolved cases, it was found that a small amount of smoothing applied to the computed Hessian of φ improves robustness and accuracy. To perform this smoothing, the discrete Hessian is input to a partition of unity algorithm, as discussed further in §Appendix B; the resulting functional is used to evaluate the lifting terms for the interfacial jump data in the backward Euler/Crank–Nicolson solve for $\mathbf{m}$.

The presented predictor–corrector schemes for the above two interfacial gauge methods involve two Poisson solves per time step (for q), two projection solves per time step (to compute $\mathbf{u}$ from $\mathbf{m}$), and two viscous solves per time step (for $\mathbf{m}$). By using the multigrid preconditioned conjugate gradient algorithms presented earlier in this work [1], all three types of solvers have computational complexity $\mathcal{O}(N)$ (ignoring parallelisation), where N is the total number of mesh elements. Generally speaking, the projection solves take the most time per time step,[2] which is typical of incompressible fluid flow solvers. It may also be possible to apply other types of temporal schemes to interfacial gauge methods, such as multi-step methods analogous to those used by consistent splitting schemes [15] and slip correction schemes [16], as well as high-order accurate spectral deferred corrections [17]. As part of such an investigation, different schemes for defining q could also be evaluated, e.g., ones which avoid solving a Poisson problem, as mentioned in §2.2.1, but for simplicity of presentation, these ideas are not pursued further as part of the present work.

2.2.3. Convergence tests

Since there are no known nontrivial test problems with exact solutions for surface tension dynamics, we consider a series of grid convergence studies to examine the spatial and temporal order of accuracy. (Some of the following convergence tests

[2] As an example, the high resolution capillary wave dynamics example given in the left column of Fig. 11 involved approximately 16,000 bicubic elements, 2,000 time steps, and was computed in approximately 5 minutes using 16 nodes on NERSC's Edison Cray XC30; approximately 30% of the time was spent in the projection solve, 15% of the time in the viscous solve, 15% in the q solve, 10% in recomputing discrete differential operators as the mesh evolves, 10% in applying partition of unity algorithms, and the remainder of the time spread across other components of the framework.

first appeared in [8] and are repeated here for completeness; however, the cited work did not present temporal convergence tests nor examine conservation of mass, as is done here.) In all cases, we consider an example in which the dynamics is mildly resolvable on a relatively coarse 8×8 background grid in 2D ($8 \times 8 \times 8$ grid in 3D) using the lowest-order elements. The test problem is chosen such that the background grid can be uniformly refined for a handful of levels, yet remain tractable in three dimensions with high-order elements. The problem consists of an interface, initially a sine-wave of one wavelength, separating two fluids in a rectangular box. Specifically,

- In two dimensions, the domain is $\Omega = (-\frac{1}{2}, \frac{1}{2})^2$ and the initial interface is given by the zero level set of $\phi(x, y) = y - 0.025 \cos 2\pi x$. Flow is periodic in the x-direction, and there are no-slip walls on the bottom and top boundaries $\{y = \pm\frac{1}{2}\}$.
- In three dimensions, the domain is $\Omega = (-\frac{1}{2}, \frac{1}{2})^3$ and the initial interface is given by the zero level set of $\phi(x, y, z) = y - 0.025(0.6 + 0.4 \cos 2\pi z) \cos 2\pi x$. Therefore, the interface is a single sine wave in the x-direction with amplitude modulated by a sine wave in the z-direction. Flow is periodic in the x and z directions, and there are no-slip walls on the bottom and top boundaries $\{y = \pm\frac{1}{2}\}$.

In one set of tests, "Test Case A," both phases have the same density $\rho = 1$ and same viscosity $\mu = 1$, employing the interfacial gauge method (4)–(6); in the second set of tests, "Test Case B," one phase has $\rho = 1$ and $\mu = 1$, whereas the other phase has $\rho = \frac{1}{16}$ and $\mu = \frac{1}{8}$, employing the interfacial gauge method (7)–(9). In all cases, surface tension is set equal to $\gamma = 10$. These parameters are chosen such that surface tension forces are relatively strong and the ensuing dynamics are mildly resolvable on a coarse grid; at the final time considered below and for both test cases, the interface is moving with typical speed $U = 0.2$ over a length scale $L = 1$, and so, relative to the heaviest fluid, the Reynolds number $\mathsf{Re} = \rho U L/\mu$ is of the order 0.2, the Capillary number $\mathsf{Ca} = \mu U/\gamma$ is of the order 0.02, and the Weber number $\mathsf{We} = \rho U^2 L/\gamma$ is of the order 0.004. To initialise the interfacial gauge methods, in their respective time stepping algorithms we set $\mathbf{m}^0 = \mathbf{u}^0 = 0$ and $\varphi^0 = \varphi_t^{-1/2} = \varphi_t^{-3/2} = 0$.

To assess convergence, a reference solution is calculated with the finest mesh (smallest h) and highest-order elements (largest p). Naturally, results computed on different meshes are likely to yield different interface positions. Because none of the state quantities are smooth across the interface (most notably pressure, which is generally discontinuous), one must therefore be careful to define a metric taking this aspect into account. The metric used in the following employs the maximum norm, except that any points x in the domain in which two simulations disagree on the phase identifier of x are excluded. Specifically, let f and $\tilde{f}$ be the same quantity (e.g., pressure) computed on two different meshes, and let Ω_i be the set of points in phase $i = 1, 2$ on the first mesh and $\tilde{\Omega}_i$ the set of points in phase i on the second mesh. Then define the metric

$$e(f, \tilde{f}) := \sup_{x \in (\Omega_1 \cap \tilde{\Omega}_1) \cup (\Omega_2 \cap \tilde{\Omega}_2)} \left| f(x) - \tilde{f}(x) \right|.$$

Together with the knowledge that the interface position is converging with high-order accuracy (implying the width of the region $(\Omega_1 \cap \tilde{\Omega}_2) \cup (\Omega_2 \cap \tilde{\Omega}_1)$ is vanishingly small), this metric effectively examines convergence, in the infinity norm, throughout the domain and thus also next to the interface. To measure convergence of the interface itself, we here make use of the fact that the norm of the gradient of the level set function ϕ is bounded away from zero, and so if $e(\phi_h, \phi_{\text{exact}}) = \mathcal{O}(h^q)$, then the interface is converging with (at least) qth-order accuracy.

Temporal order of accuracy. To examine the temporal order of accuracy, a reference solution for Test Cases A and B is computed using a 32×32 background Cartesian grid ($h = 2^{-5}$) with $p = 4$ biquartic elements in 2D (empirically determined to yield spatial errors of negligible size compared to temporal errors), using a time step of $\Delta t \approx 2.4 \times 10^{-6}$. The above defined metric is then used to measure the maximum-norm error in the computed solution at time $t = 0.01$, for a sequence of discrete solutions computed with progressively larger time steps. The results are shown in Fig. 6 and show that all state quantities attain second-order temporal accuracy.

Spatial order of accuracy. To study the spatial order of accuracy, a reference solution is computed on the finest mesh (smallest h) and highest-order elements (largest p) with a fixed time step of $\Delta t = 10^{-5}$. In addition to using the maximum norm in space, the maximum norm in time over the range $0 \le t \le T$ is also used; in two dimensions, $T = 0.01$, whereas in three dimensions, for reasons of limited computing resources, a shorter time horizon is used, $T = 0.001$. In 2D, we consider background Cartesian grids of size 8×8 to 64×64 together with $p = 2, 3$, and 4, corresponding to biquadratic, bicubic, and biquartic elements. In 3D, the grid sizes range from $8 \times 8 \times 8$ to $32 \times 32 \times 32$, with triquadratic, tricubic, and triquartic elements. Fig. 7 and Fig. 8 collect the results, showing the maximum-norm error for the fluid velocity, pressure, and level set function, for the two-dimensional and three-dimensional results, respectively.

The 2D and 3D results draw similar conclusions: for $p = 2$ and $p = 3$, approximately second-order accuracy is obtained in all quantities, whereas for $p = 4$, fourth-order accuracy is obtained. This dependence on the parity of p is directly due to the numerical method for evaluating interface curvature κ, described further in Appendix C. Recall that for tensor-product degree $(p+1)^d$ polynomials, the optimal order of accuracy one may expect in a dG method is $p + 1$. Because curvature is a second-order differential, up to two orders of accuracy compared to the optimal may be lost, which our results show is the case for $p = 3$. Therefore, it is favourable that $p = 2$ yields second-order accuracy, and $p = 4$ yields fourth-order accuracy. Nevertheless, note that high-order accuracy in velocity and pressure is obtained, uniformly throughout the domain—certain

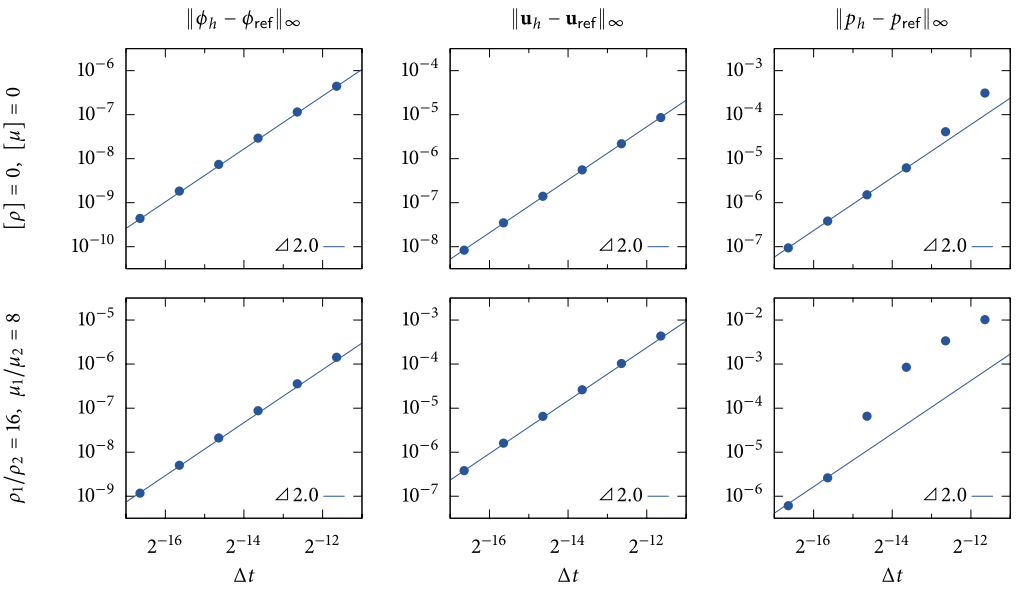

Fig. 6. Results of a temporal convergence analysis for the two-phase interfacial gauge method (4)–(6) ("Test Case A," top row) and (7)–(9) ("Test Case B," bottom row) showing the maximum-norm error in the computed velocity field, pressure, and level set function, as a function of time step Δt, comparing against the reference solution computed with a time step of $\Delta t \approx 2^{-18.6}$. Data points represent measured errors at time $t = 0.01$, whereas the lines of indicated slope illustrate the observed order of accuracy.

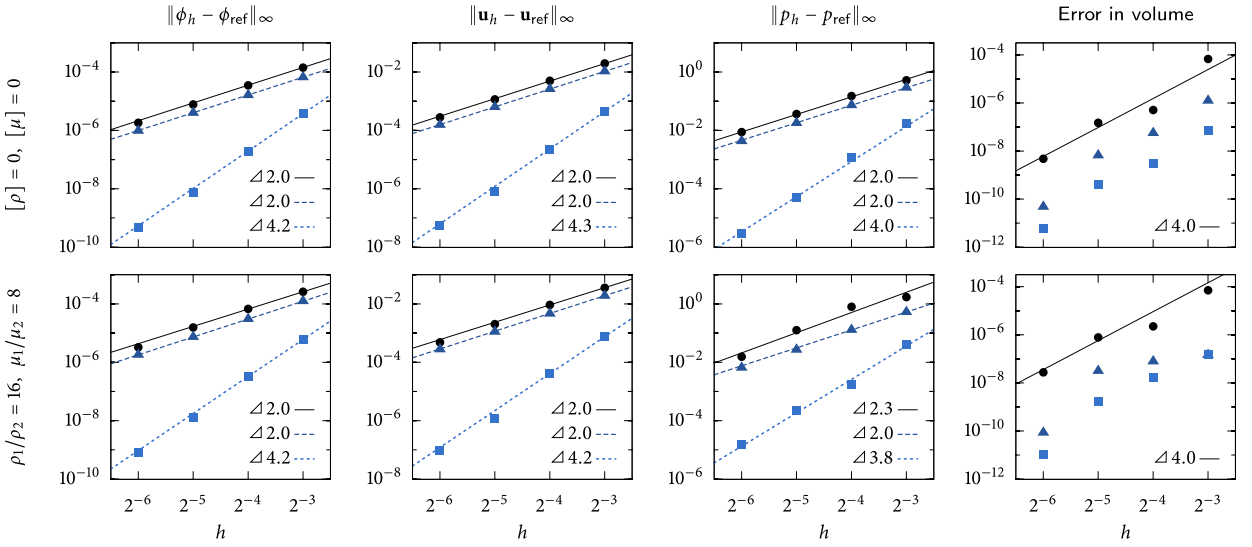

Fig. 7. Two-dimensional results of a spatial order of accuracy convergence analysis for the two-phase interfacial gauge method (4)–(6) ("Test Case A," top row) and (7)–(9) ("Test Case B," bottom row) showing the maximum-norm error in the computed velocity field, pressure, and level set function, as a function of typical element size h and order p, comparing against a reference solution computed on a 128×128 grid ($h = 2^{-7}$) with $p = 4$ elements. Also indicated is the error in volume, measured at time $T = 0.01$. Data points represent measured errors, whereas the lines of indicated slope illustrate the observed order of accuracy. • denotes $p = 2$, ▲ denotes $p = 3$, ■ denotes $p = 4$.

types of projection methods, especially those involving regularised interfacial forces using smoothed delta functions, give rise to numerical boundary layers (sometimes referred to as "parasitic currents"), often leading to first order accuracy in the maximum norm. In contrast, the combination of sharp interface methods (i.e., the implicit mesh dG framework), as well as the design of interfacial gauge methods (see [8]), allow for sharp resolution of dynamics immediately adjacent to the interface. Figs. 9 and 10 provide further insights into the two test problems, the meshes involved, and the qualitative behaviour of the discrete solutions: depicted in the figures is the implicitly defined mesh for the 2D test problem, based on a background 8×8 Cartesian grid, together with the computed results for $p = 4$ biquartic elements and the associated pointwise error. One can observe that the error is discontinuous across elements; note also that the (physical) discontinuity in pressure is sharply resolved.

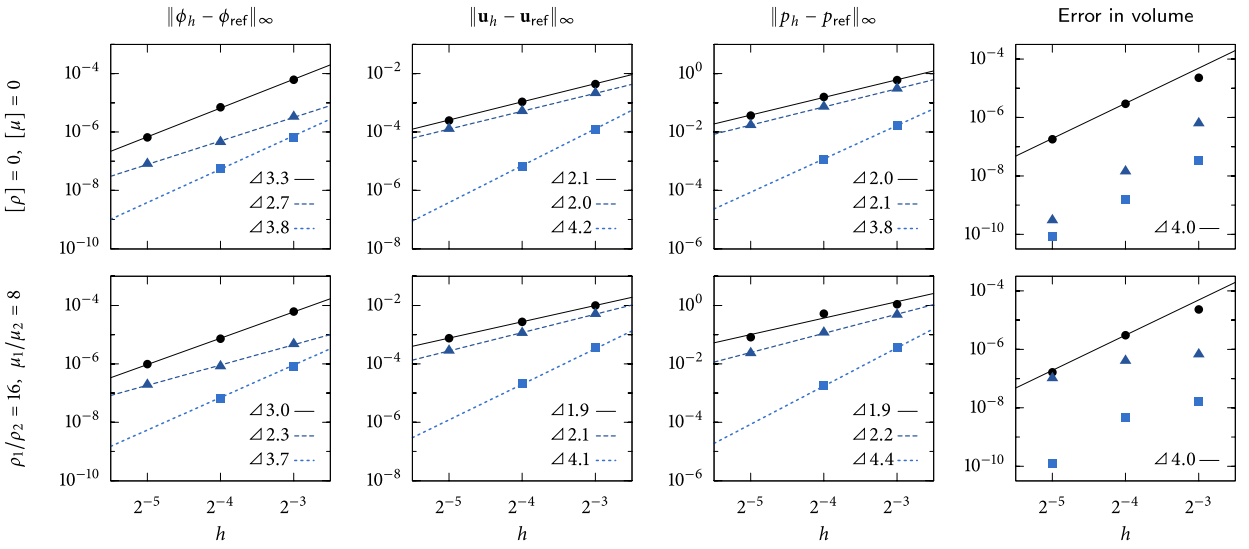

Fig. 8. Three-dimensional results of a spatial order of accuracy convergence analysis for the two-phase interfacial gauge method (4)–(6) ("Test Case A," top row) and (7)–(9) ("Test Case B," bottom row) showing the maximum-norm error in the computed velocity field, pressure, and level set function, as a function of typical element size h and order p, comparing against a reference solution computed on a 32×32 grid ($h = 2^{-5}$) with $p = 4$ elements. Also indicated is the error in volume, measured at time $T = 0.001$. Data points represent measured errors, whereas the lines of indicated slope illustrate the observed order of accuracy. • denotes $p = 2$, ▲ denotes $p = 3$, ■ denotes $p = 4$.

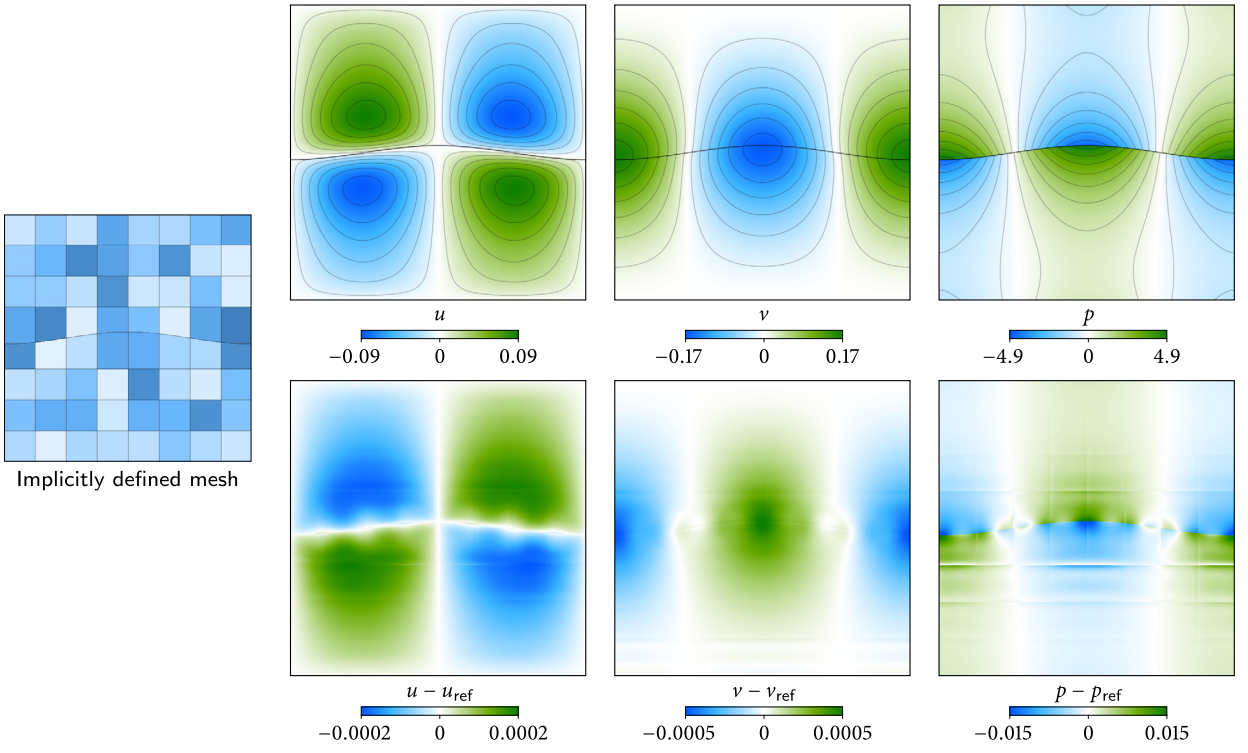

Fig. 9. Corresponding to the spatial order of accuracy grid convergence study of §2.2.3, wherein $[\rho] = 0$ and $[\mu] = 0$, in the specific case of an 8×8 background Cartesian grid and $p = 4$ biquartic elements, the discrete solution is shown at time $t = 0.01$. The top row illustrates the computed velocity field $\mathbf{u} = (u, v)$ and pressure field p, which are each piecewise polynomial functions on the depicted implicitly defined mesh; the bottom row plots the error in this discrete solution, which exhibits discontinuities in alignment with the element boundaries.

Another error metric frequently used to examine convergence in two-phase fluid dynamics is that of conservation of volume. Although such a metric is very weak, as it has a large degree of error cancellation and often indicates a delusory convergence rate, for completeness it is included in Figs. 7 and 8. The asymptotic convergence rate is unclear, as the sign

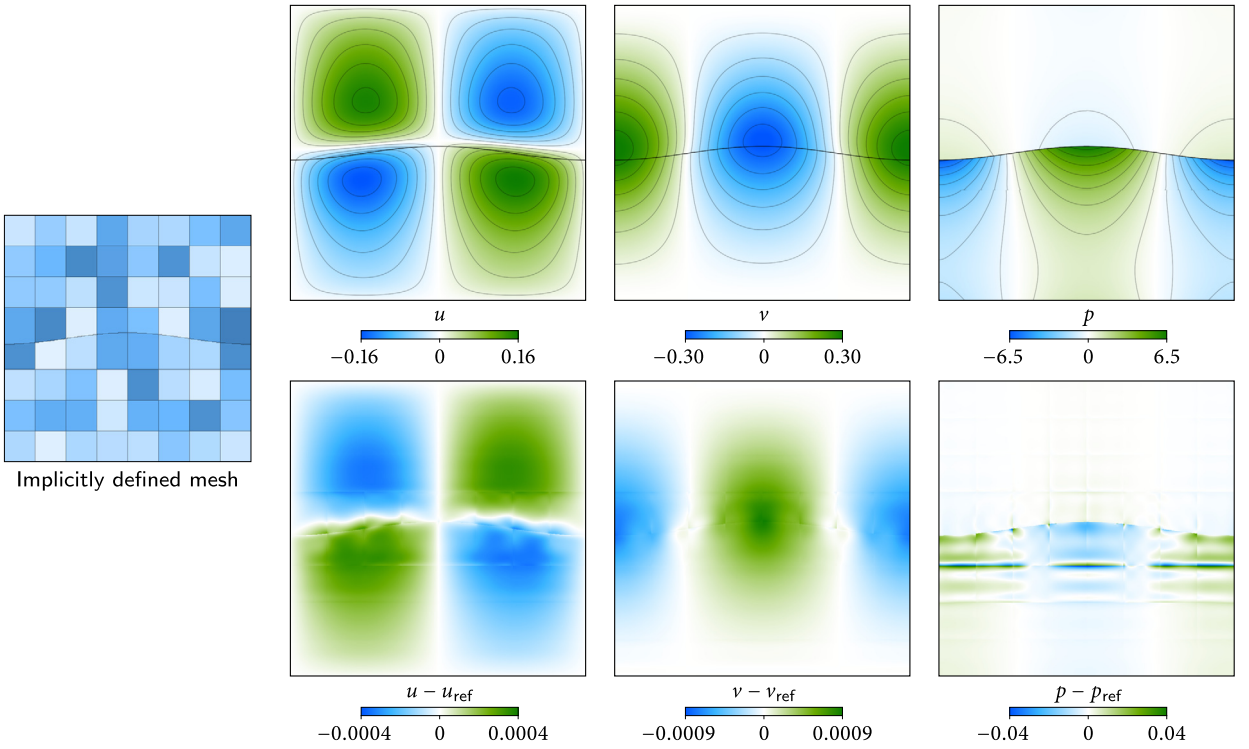

Fig. 10. Corresponding to the spatial order of accuracy grid convergence study of §2.2.3, wherein $\rho_1/\rho_2 = 16$ and $\mu_1/\mu_2 = 8$, in the specific case of an 8×8 background Cartesian grid and $p = 4$ biquartic elements, the discrete solution is shown at time $t = 0.01$. The top row illustrates the computed velocity field $\mathbf{u} = (u, v)$ and pressure field p, which are each piecewise polynomial functions on the depicted implicitly defined mesh; the bottom row plots the error in this discrete solution, which exhibits discontinuities in alignment with the element boundaries.

of the error metric frequently changes, being highly-dependent on mesh topology. Nevertheless, the indicative convergence rate for volume conservation is at least equal to that of the level set function and is sometimes higher (for example, the second order $p = 2$ method has fourth-order accurate conservation of mass), and shows that volume is conserved very well by the combined level set method, implicit mesh dG, and interfacial gauge method framework.

As an additional means of validation, results of the presented framework have also been compared to that of a second-order accurate, purely finite difference-based "jump splice" methodology [24]. The comparison consisted of two problems involving an ellipsoidal interface oscillating under the influence of surface tension: in one test case, the Reynolds number was 10 and the bubble dampens to a circle after a couple of oscillations; in the second case, the Reynolds number is 100 and the bubble oscillates several times with a small amount of damping. Using the highest resolution results of [24] (grid size 1024×1024) as a reference solution, and again using a maximum norm metric, convergence rates similar to the above were observed in all quantities (velocity, pressure, and interface location), up to the limited accuracy of the finite difference solution; results are not presented here.

2.2.4. *Capillary wave dynamics*

In [8], two examples are given of surface tension dynamics giving rise to intricate fine-scale flow features. Since these examples serve to illustrate a variety of phenomena also exhibited by other examples elsewhere in this work, they are repeated here for completeness. Sharing the same initial condition, the two examples each consist of two fluids separated by an interface resembling a soap bubble shortly after coalescing with a flat film; see the $t = 0$ frames of Fig. 11. In the first case, utilising the interfacial gauge method in (4)–(6), both phases have the same density $\rho = 1$ and viscosity $\mu = 0.001$; in the second case, utilising the interfacial gauge method in (7)–(9), the fluid on top has a density of $\rho = 1$ and a viscosity of $\mu = 0.001$, whereas the fluid on the bottom has a density of $\rho = 10$ and a viscosity of $\mu = 0.01$. In both cases, the surface tension coefficient is $\gamma = 0.1$, no-slip boundary conditions are imposed on the top and bottom horizontal boundaries, and the flow is periodic in the horizontal direction and is initially stationary, $\mathbf{u} \equiv 0$ at $t = 0$. Computed with $p = 3$ bicubic polynomial elements, using a background Cartesian grid of size 128×48, Fig. 11 illustrates the resulting dynamics, wherein the surface tension of the initial "bubble" causes it to collapse, generating capillary waves in the process. The figures also plot the vorticity $\omega = \nabla \times \mathbf{u} = \partial_x v - \partial_y u$ at select moments in time, and reveal intricate "heart"-shaped features aligning with the wavelengths of the capillary waves. In particular, in the first case the vorticity is continuous across the interface (because $[\mu] = 0$ and so $[\nabla \mathbf{u}] = 0$, see [8]), whereas in the second case, the vorticity is generally discontinuous across the interface. These discontinuities are sharply captured by the implicit mesh dG method and are high-order accurate owing

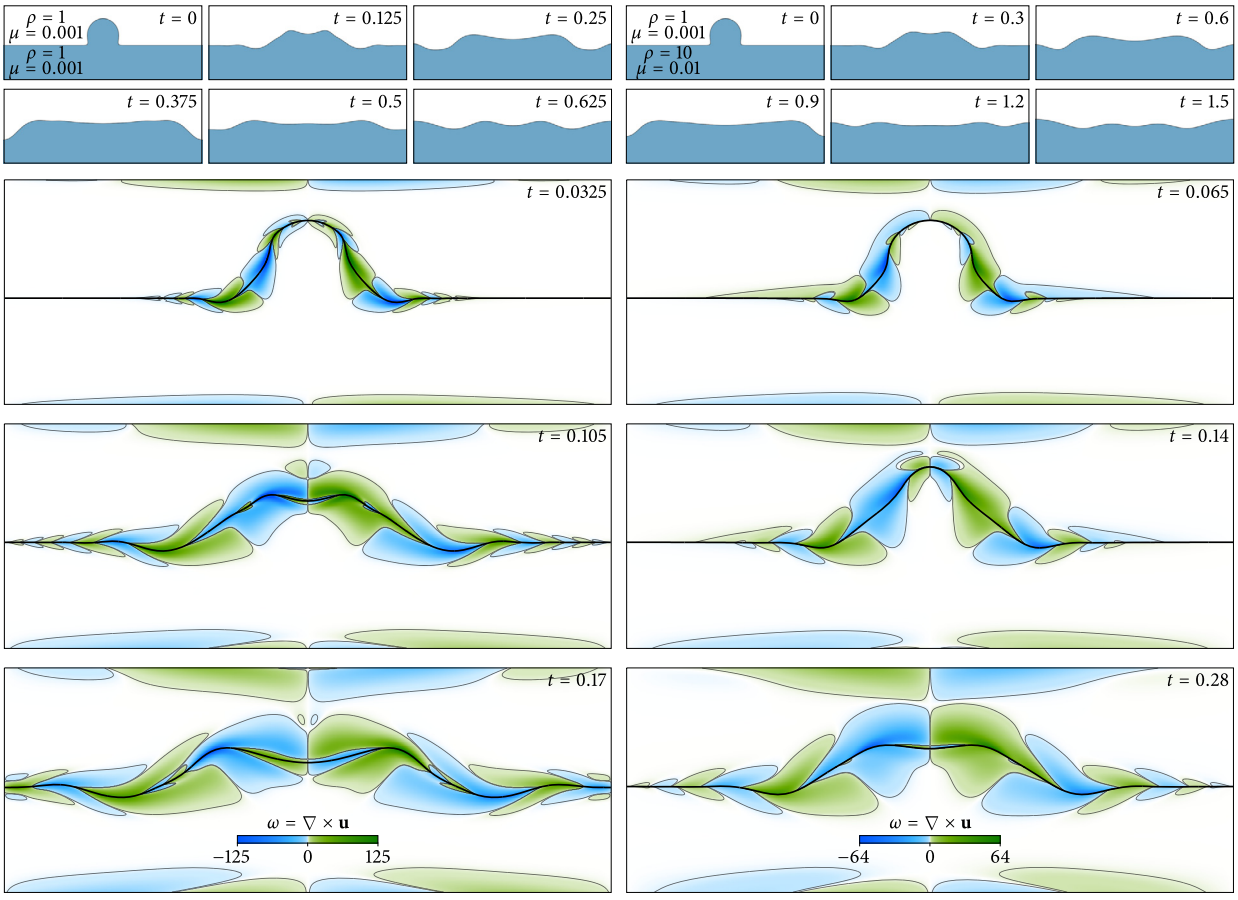

Fig. 11. Capillary waves in surface tension dynamics (adapted from [8]). *(left)* An example in which $[\rho] = 0$ and $[\mu] = 0$. *(top panel)* Initial condition together with a sample of frames to provide an indication of the resulting wave dynamics. *(bottom panel)* A plot of vorticity $\omega = \nabla \times \mathbf{u}$ at three moments in time – the "heart" shaped features correspond precisely with the wavelengths of the capillary waves. Note also the vorticity generation at the no-slip walls near the top and bottom. *(right)* An example involving a jump ratio of 10 in both density and viscosity. *(top panel)* Initial condition together with a sample of frames to provide an indication of the resulting wave dynamics. *(bottom panel)* A plot of vorticity $\omega = \nabla \times \mathbf{u}$ at three moments in time – similar to the left case, there are "heart" shaped features corresponding to the wavelengths of the capillary waves, but in this case the vorticity is discontinuous across the interface.

to the design of interfacial gauge methods. Using a maximum fluid speed of $\|\mathbf{u}\| \approx 1.8$ (respectively, 0.6) and the length scale of the domain, $L = 1$, the Reynolds number $\mathsf{Re} = \rho U L/\mu$ for the flow in the left-half of Fig. 11 is approximately 1800 (respectively, 600 for the right half), while, relative to the density and viscosity of the top fluid, the Capillary number $\mathsf{Ca} = \mu U/\gamma$ is approximately 0.02 (resp., 0.006), and the Weber number $\mathsf{We} = \rho U^2 L/\gamma$ is approximately 32 (resp., 4).

2.2.5. Soap bubble oscillation dynamics

In the next example, we consider a three-dimensional bubble oscillation problem and compare a highly-resolved computation using the presented two-phase Navier–Stokes solver with that of an experiment conducted by Kornek et al. [25]. When two spherical soap bubbles collide and merge, surface tension causes the resulting larger bubble to oscillate rapidly; over time, the oscillations dampen, owing to the viscous damping of the gas inside and outside the bubble (and also, to some extent, from the viscous damping of the soapy liquid within the thin film), eventually leading to a bubble spherical in shape. In the work of Kornek et al., high-speed movies of bubble oscillations are used to assess the validity of a various models predicting the frequency and damping rate of the different modes of oscillation as function of fluid density, viscosity, and surface tension. We use one of these experiments here to compare with the two-phase Navier–Stokes solver presented in this work, serving as a form of experimental validation for the presented interfacial gauge methods for surface tension dynamics.

Based on movie "08" of [25], two protogenic spherical bubbles of radius ≈1.78 cm and ≈2.31 cm, overlapping by 3 mm, are defined as the initial condition; see the $t = 0$ frame of Fig. 13. Values for density and surface tension are defined by those quoted in [25], i.e., $\rho = 1.2\ \mathrm{kg\,m^{-3}}$ and $\gamma = 0.068\ \mathrm{kg\,s^{-2}}$. As discussed in Kornek et al., experimental results consistently indicate that soap bubbles dampen more quickly than as predicted by idealised theoretical and numerical models, wherein one considers only the viscous damping of the gas. This difference is attributed to the influence of the non-zero film

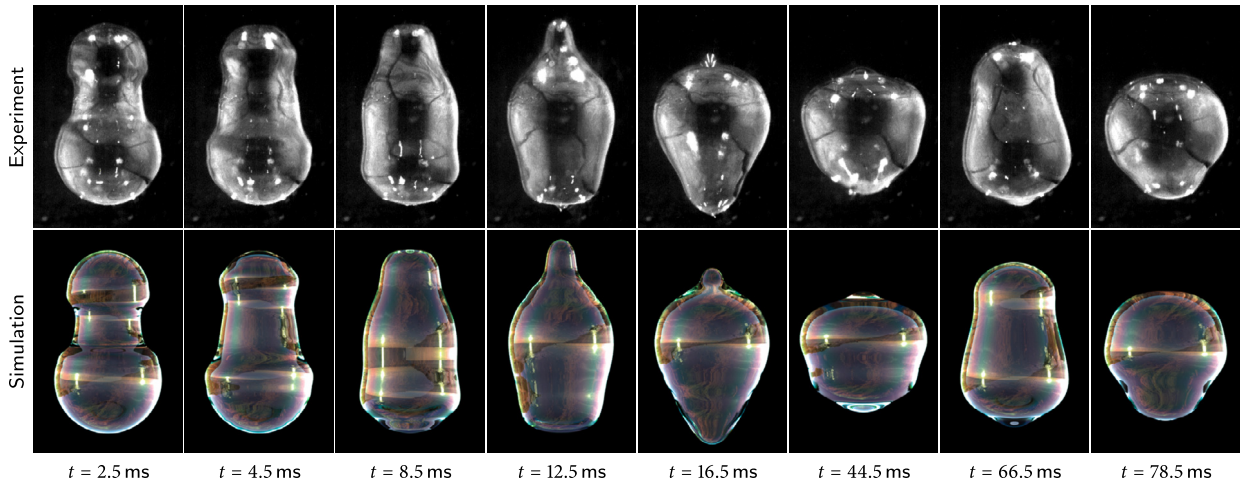

Fig. 12. Comparison of numerical results (bottom row, computed with the implicit mesh dG framework) with that of a bubble oscillation experiment (top row, reproduced with permission from Kornek et al. [25]) in which two spherical bubbles collide and merge at $t = 0$, subsequently causing oscillations driven by surface tension (see Fig. 12 for the initial condition). The numerical results are computed with an axisymmetric formulation of the dG framework, and are visualised with a ray-traced rendering using thin-film interference.

thickness and the loss of energy due to viscous damping of the liquid in the film, whereas the theoretical models and numerical experiments often assume a vanishingly thin interface. In fact, using the quoted value of viscosity of the gas used in the experiments (1.8×10^{-5} kg m^{-1} s^{-1}), simulations conducted in the present work revealed appreciable overshoot in the ensuing bubble oscillations, clearly in disagreement with experimental results. Since we do not attempt to model the non-zero thickness of the film in this work, and instead consider an idealised model wherein the interface is vanishingly thin, in the following we instead set viscosity to be twice the experimental value, i.e., $\mu = 3.6 \times 10^{-5}$ kg m^{-1} s^{-1}. This compensating factor of two is in alignment with that observed elsewhere [25].

To take advantage of the axisymmetry present in this problem, the implicit mesh dG framework can be reformulated in cylindrical coordinates. In general, the only required modifications are to include a radial Jacobian factor within all of the mass matrices, with a few exceptions relating to the vector Laplacian and to calculating mean curvature of the interface, see §Appendix A. Even with the reduced computational complexity that follows (compared to a fully-three dimensional simulation), due to the small viscosity of the gas, it is nevertheless advantageous to use adaptive mesh refinement to correctly resolve the small boundary layers next to the evolving interface. The mesh refinement strategy employed here is a straightforward one: coarse cells in the background quadtree are based on a 16×32 grid, with a gradual refinement (based on distance to the interface) to the finest cells which are 32 times smaller (i.e., six levels); see Fig. 13. In order to follow the evolving interface, the mesh is re-adapted approximately every few hundred time steps. The domain is $\Omega = \{(r, z) : 0 < r < 6, -6 < z < 6\}$, suitably large enough to envelop the bubble system; boundary conditions at $r = 0$ are those naturally required by axisymmetry; at $r = 6$, no-slip boundary conditions are imposed, while for $z = \pm 6$, periodic boundary conditions are used, though none of the latter boundary conditions are expected to have a significant influence on the bubble dynamics.

Figs. 12, 13, and 14 illustrate the results of the axisymmetric bubble oscillation computation, which is computed with the implicit mesh dG framework using $p = 3$ bicubic elements, together with the interfacial gauge method (4)–(6) implemented by Algorithm 2. Fig. 12 compares the numerical results with that of the high-speed experiment movie at a selection of frames, showing good agreement between the two. Fig. 13 examines the results in more detail by plotting a radial slice of the gas speed $\|\mathbf{u}\|$ and pressure p. Note the exaggerated colour scale for the gas speed $\|\mathbf{u}\|$, which is needed to highlight the small boundary layers next to the interface. With a velocity scale about equal to the maximum speed, $U \approx 14$ m s^{-1}, and the length scale of the domain, $L = 6$ cm, the maximum Reynolds number $\mathrm{Re} = \rho U L/\mu$ for this flow is approximately 28,000 (the Capillary number $\mathrm{Ca} = \mu U/\gamma$ is approximately 0.007, and the Weber number $\mathrm{We} = \rho U^2 L/\gamma$ is approximately 200). Clearly, one may therefore expect turbulent flow within a fully three-dimensional oscillating bubble over the rapidly oscillating time scales considered; however, turbulence is not observed within the numerical results because of the imposed axisymmetry and steady state initial condition, $\mathbf{u} \equiv 0$ at $t = 0$. Fig. 13 also shows the computed surface area of the bubble and its volume as a function of time. The model being considered here is the two-phase incompressible Navier–Stokes equations, with no permeability at the interface, thus the volume of the bubble should theoretically be conserved; the figure shows that it is conserved very well (deviating by one-hundredth of 1%) by the numerical method, which makes no attempt to conserve mass exactly in the discrete setting. The second plot shows a trend of surface area reduction, as one may expect as viscosity dampens the oscillations into a spherical bubble shape, with a lower limit of $\approx$86.3 cm^2 for the considered bubble having volume $\approx$75.3 cm^3. Last, Fig. 14 plots the azimuthal component of the vorticity $\nabla \times \mathbf{u}$ at three moments in time: in the first frame, one can observe the "heart"-shaped structures that were observed in the previous example on capillary wave dynamics (§2.2.4); the second frame, occurring shortly after the tip of the bubble inverts (see

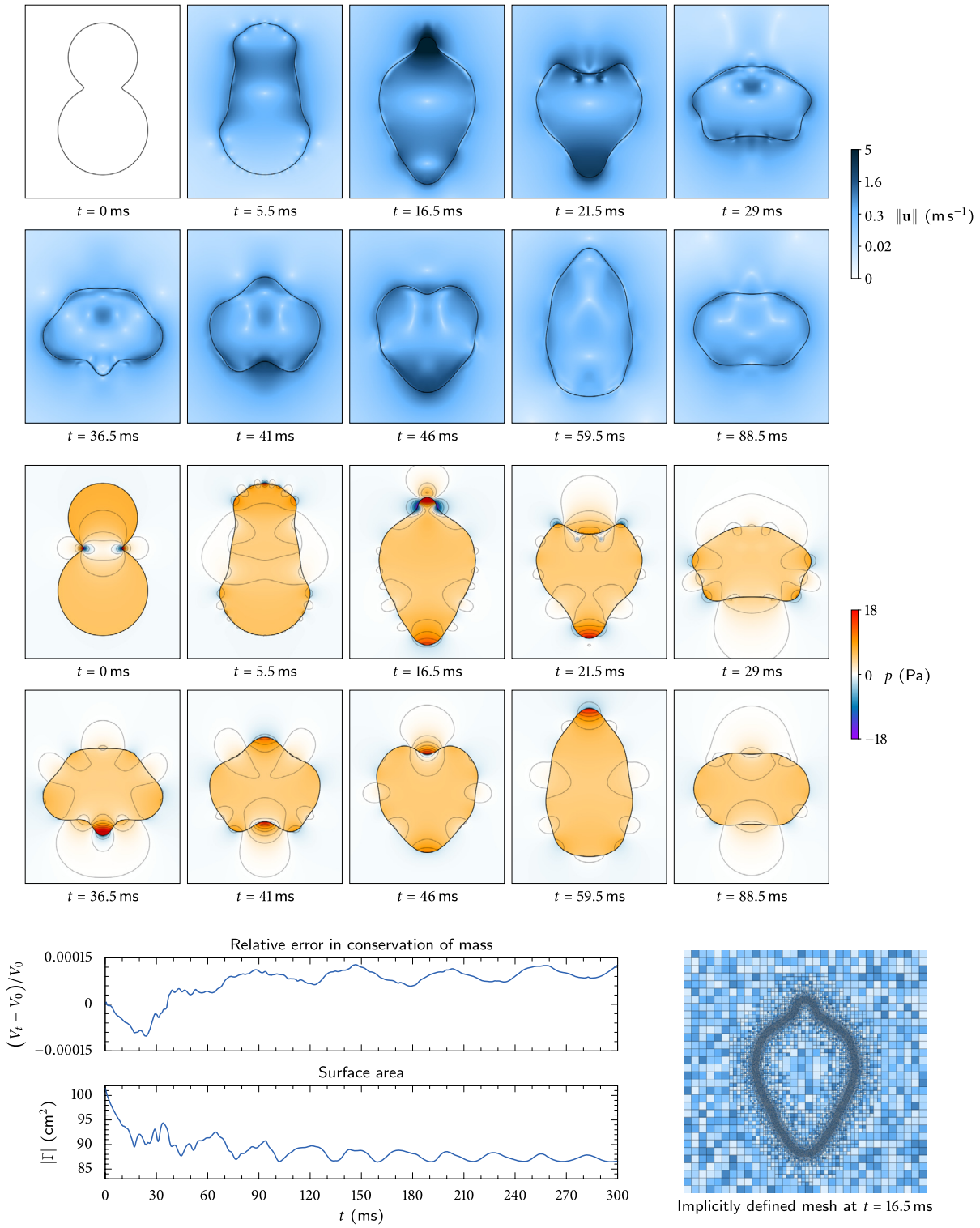

Fig. 13. Computed results of the axisymmetric bubble oscillation problem considered in §2.2.5 (see also Fig. 12). (*top panels*) Plots of the speed $\|\mathbf{u}\|$ and pressure p of the gas inside and outside the bubble at various moments in time (bubble surface indicated by thick black curve). Note the exaggerated scale for the gas speed, which is needed to highlight the wide-ranging velocity scales involved and the small boundary layers next to the interface. (*bottom left panel*) Plots of the bubble volume and surface area as a function of time, showing a deviation of at most 0.015% in the discrete bubble volume, and a trend to reduce surface area as the bubble dampens into a spherical shape. (*bottom right*) Illustration of the adaptively refined mesh, consisting of six levels of refinement in the quadtree; the refinement adapts to the interface as it evolves in time. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

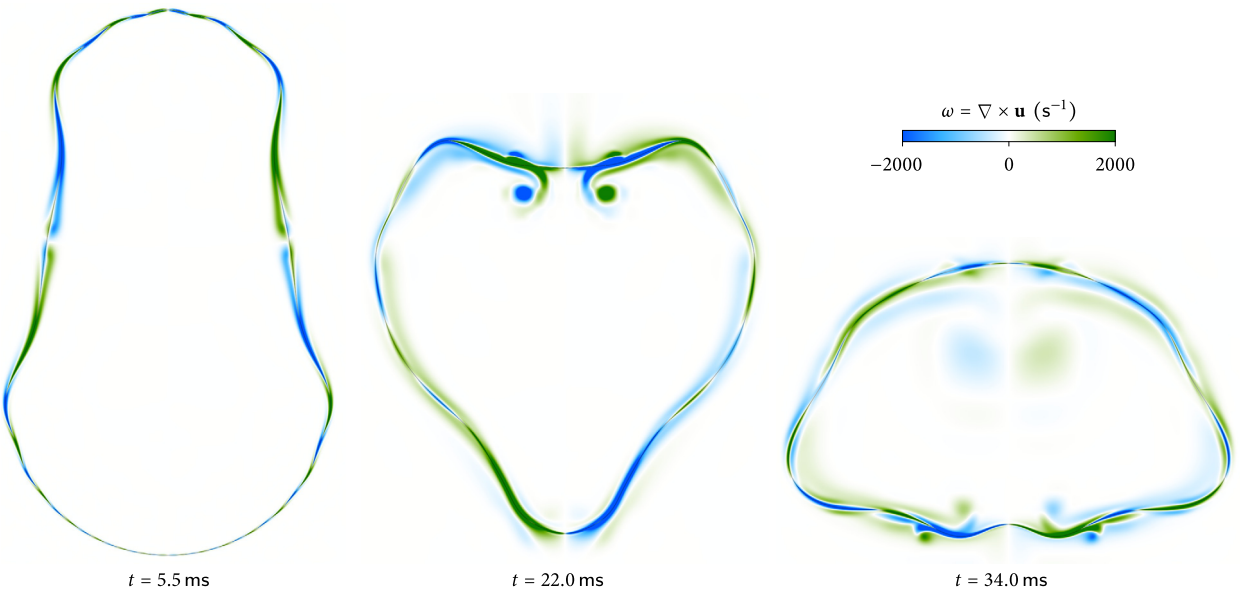

Fig. 14. Corresponding to the axisymmetric bubble oscillation problem considered in §2.2.5 (see also Fig. 13), plots of the azimuthal component of the gas vorticity at three moments of interest. Note the "heart"-shaped features, similar to those seen in the capillary wave dynamics example in §2.2.4, as well as the vortex generation (in fact, vortex tubes, by axisymmetry) as the bubble rapidly oscillates.

the $t = 16.5$ ms frame of Fig. 13), shows the generation of vortices that subsequently travel into the interior of the bubble; remnants are visible in the third frame. As the numerical results show, there is a wide breadth of length scales in this bubble oscillation problem: for example, the boundary layers next to the interface are a fraction of a millimetre thick, whereas the radius of the bubble is a few centimetres. Depending on the modelling question at hand, it may or may not be necessary to resolve these boundary layers: if one is interested only in oscillation frequency, say, then precise resolution may be unnecessary; if on the other a careful analysis of damping dynamics is being considered, or the effect fluid stress has on film dynamics is of interest, then modelling these boundary layers is likely necessary.

2.3. Rigid body fluid–structure interaction

Interfacial gauge methods can also be applied to fluid–structure interaction, as demonstrated here with an application to modelling the motion of a rigid body embedded in an incompressible fluid. We adopt here and expand upon the interfacial gauge method first outlined in [8], describing a time stepping method and its implementation within the implicit mesh dG framework.

2.3.1. Equations of motion

Consider a rigid body Ω_b surrounded by a fluid of density ρ and viscosity μ. The motion of the body is determined by the net force and torque exerted on the body by the fluid, which, in turn, is determined by integrating the normal component of the fluid stress $\boldsymbol{\sigma}$ around the body surface $\Gamma := \partial\Omega_b$. Since the body is rigid, the velocity field inside the body satisfies $\mathbf{u}_b = \mathbf{u}_b(x) = v + \omega \times (x - x_c)$, where v is the body's linear velocity, ω its angular velocity, and x_c its centre of mass. (Here, and in most of the following, we assume a three-dimensional problem; in the case of two dimensions, cross products and moments of inertia simplify in the natural way.) The body's position and velocity vectors evolve according to conservation of linear and angular momentum, such that

$$\begin{aligned} \dot{x}_c &= v, \\ m\dot{v} &= \textstyle\int_\Gamma \boldsymbol{\sigma}\mathbf{n} + mg, \\ \tfrac{d}{dt}(I_b\omega) &= \textstyle\int_\Gamma (x - x_c) \times \boldsymbol{\sigma}\mathbf{n}, \end{aligned}$$

where $m = \int_{\Omega_b} \rho_b$ is the body's mass, ρ_b is the body's density (which may be non-uniform), $x_c = m^{-1}\int_{\Omega_b} x\,\rho_b\,dx$ is its centre of mass, $I_b = \int_{\Omega_b} \rho_b(x)\big(|x - x_c|^2\mathbb{I} - (x - x_c)(x - x_c)^\intercal\big)\,dx$ is its moment of inertia, g is gravity, and $\mathbf{n}$ is the unit normal on Γ pointing into the fluid. The fluid in the domain $\Omega_f := \Omega \setminus \Omega_b$ solves the Navier–Stokes equations with gravitational forcing, subject to the boundary condition $\mathbf{u} = \mathbf{u}_b$ on Γ,

$$\begin{cases} \rho(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}) = -\nabla p + \mu\Delta\mathbf{u} + \rho g & \text{in } \Omega_f \\ \nabla\cdot\mathbf{u} = 0 & \text{in } \Omega_f \\ \mathbf{u} = \mathbf{u}_b & \text{on } \Gamma \\ \mathbf{u} = \mathbf{u}_\partial & \text{on } \partial\Omega_f \setminus \Gamma. \end{cases} \tag{12}$$

An interfacial gauge method which stably and accurately solves this system of equations is as follows [8]: let $\mathbf{m} : \Omega_f \to \mathbb{R}^d$, $\varphi : \Omega_f \to \mathbb{R}^d$ and $q : \Omega_f \to \mathbb{R}^d$ solve

$$\left.\begin{array}{c} \rho(\mathbf{m}_t + \mathbf{u}\cdot\nabla\mathbf{u}) = -\nabla q + \mu\Delta\mathbf{m} \\ \mathbf{u} = \mathbf{m} - \nabla\varphi \\ \Delta\varphi = \nabla\cdot\mathbf{m} \\ \Delta q = 0 \end{array}\right\} \text{ in } \Omega_f, \tag{13}$$

subject to the interface conditions

$$\left.\begin{array}{c} \mathbf{m}\cdot\mathbf{n} = \mathbf{u}_b\cdot\mathbf{n} \\ \mathbf{m}\cdot\boldsymbol{\tau} = \mathbf{u}_b\cdot\boldsymbol{\tau} + \nabla\varphi\cdot\boldsymbol{\tau} \\ \partial_n\varphi = 0 \\ \partial_n q = -\rho(\dot{v} + \dot{\omega}\times(x-x_c) + \omega\times(\omega\times(x-x_c)))\cdot\mathbf{n} \end{array}\right\} \text{ on } \Gamma, \tag{14}$$

and domain boundary conditions

$$\left.\begin{array}{c} \mathbf{m}\cdot\mathbf{n} = \mathbf{u}_\partial\cdot\mathbf{n} \\ \mathbf{m}\cdot\boldsymbol{\tau} = \mathbf{u}_\partial\cdot\boldsymbol{\tau} + \nabla\varphi\cdot\boldsymbol{\tau} \\ \partial_n\varphi = 0 \\ \partial_n q = 0 \end{array}\right\} \text{ on } \partial\Omega_f \setminus \Gamma. \tag{15}$$

As in the case of interfacial gauge methods for two-phase flow—wherein q represents the force of surface tension on the bulk fluid—in the above, the role of q is to implement a fluid body force corresponding to the force exerted on the fluid by the rigid body. To see this, consider a particle $x = x(t)$ situated on the surface of the body: its velocity is then $\dot{x} = v + \omega\times(x-x_c)$ and acceleration is $\ddot{x} = \dot{v} + \dot{\omega}\times(x-x_c) + \omega\times(\dot{x}-\dot{x}_c) = \dot{v} + \dot{\omega}\times(x-x_c) + \omega\times(\omega\times(x-x_c))$. Thus, the Neumann boundary condition for q calculates the normal component of the force exerted by the rigid body on the fluid. Observe also that the force of gravity within the Navier–Stokes equations for $\mathbf{u}$ has been absorbed into the gauge method in such a way that the evolution equation for $\mathbf{m}$ has no gravitational forcing. This absorption effectively alters the formula for computing the fluid pressure p. Last, it is straightforward to show that if $\mathbf{m}$ solves (13)–(15), then $\mathbf{u} = \mathbb{P}(\mathbf{m})$ solves (12), with pressure identified as $p = q + \rho\varphi_t - \mu\nabla\cdot\mathbf{m} + \rho G$, where $G = G(x) := g\cdot x$ is the gravitational potential.

2.3.2. Temporal discretisation

As in the case of interfacial gauge methods for two-phase flow, we develop a second-order accurate predictor–corrector scheme for rigid body fluid–structure interaction. Once again, we employ the implicit mesh dG framework—as the body is rigid, we define its shape using a fixed level set function ϕ defined in "body coordinates", and map ϕ to world coordinates with the aid of a rotation matrix R evolving in correspondence with ω, as follows. A particle of the body with position y in its stationary reference frame is mapped to world coordinates via $y \mapsto x_c + Ry$. To derive the evolution equation for R, consider a particle $x = x(t)$ with fixed body coordinate y: differentiating with respect to time implies that $\mathbf{u}_b(x) = \dot{x} = \dot{x}_c + \dot{R}y = v + \dot{R}y$ and so $\dot{R}y = \omega\times(x-x_c) = \omega\times Ry$. Since this holds for all y, it follows that $\dot{R} = S(\omega)R$, where

$$S(\omega) := \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

is the cross-product matrix.

In three dimensions, the body's moment of inertia I_b is, in general, time-dependent. This time-dependence can be isolated as follows: using the fact that R is orthogonal,[3] we have that

$$\begin{aligned} I_b &= \int_{\Omega_b} \rho_b(x)\big(|x-x_c|^2\mathbb{I} - (x-x_c)(x-x_c)^\intercal\big)\,dx = \int_y \rho_b(x_c+Ry)\big(|Ry|^2\mathbb{I} - (Ry)(Ry)^\intercal\big)\,dy \\ &= \int_y \tilde{\rho}_b(y)\big(|y|^2\mathbb{I} - Ryy^\intercal R^\intercal\big)\,dy = R\left[\int_y \tilde{\rho}_b(y)(|y|^2\mathbb{I} - yy^\intercal)\,dy\right]R^\intercal = R\tilde{I}_b R^\intercal \end{aligned}$$

where $\tilde{\rho}_b$ and $\tilde{I}_b := \int_y \tilde{\rho}_b(y)(|y|^2\mathbb{I} - yy^\intercal)\,dy$ is the body's density and moment of inertia as defined in its stationary reference frame. Also, note that

[3] Assuming R is orthogonal at $t = 0$, since $S^\intercal = -S$ and $\frac{d}{dt}(R^\intercal R) = (R')^\intercal R + R^\intercal R' = R^\intercal S^\intercal R + R^\intercal SR = 0$, it follows that R remains orthogonal for all time.

Algorithm 4 Predictor–corrector scheme for the interfacial gauge method for rigid body fluid–structure interaction, (13)–(15).

1: Define $\phi = \phi(y)$ and $\rho_b = \rho_b(y)$ in body coordinates.
2: Calculate $m = \int_{\phi(y)<0} \rho_b(y)$ and $\tilde{I}_b = \int_{\phi(y)<0} \rho_b(y)(|y|^2\mathbb{I} - yy^\intercal)$.
3: Initialise x_c^0 and R^0 at $t = 0$.
4: Define the initial implicit mesh using $x \mapsto \phi\big((R^0)^\intercal(x - x_c^0)\big)$ and adopt its differential operators.
5: Initialise $\mathbf{u}^0$, $\mathbf{m}^0$, φ^0, v^0, and ω^0 at time $t = 0$.
6: Determine an appropriate φ^{-1}, $\varphi_t^{-1/2}$, $\dot{v}^{-1/2}$ and $\dot{\omega}^{-1/2}$.
7: Calculate the initial q^0, $\dot{v}^0$, and $\dot{\omega}^0$ with an iterative procedure; see text.
8: **for** $n = 0, 1, 2, \ldots$ **do**
9: Define $\mathbf{a_u} := \nabla \cdot (\mathbf{u}^n \mathbf{u}^n)$.
10: Define $\mathbf{f}_q := \nabla q^n$ and $\mathbf{f_m} := \mu \Delta \mathbf{m}^n$.
Update the configuration of the body:
11: Calculate $x_c^{n+1} = x_c^n + \Delta t(v^n + \frac{\Delta t}{2}\dot{v}^n)$.
12: Calculate $R^{n+1} = \exp(\Delta t\, S(\omega^n + \frac{\Delta t}{2}\dot{\omega}^n))R^n$.
13: Define the new mesh—preserving the topology of the current mesh—using $x \mapsto \phi\big((R^{n+1})^\intercal(x - x_c^{n+1})\big)$.
14: Transfer all necessary quantities to the new mesh and adopt its differential operators.
Predictor step
15: Calculate a second-order accurate predictor of linear and angular body velocity: $v^\star := v^n + \Delta t\, \dot{v}^n$ and $\omega^\star := \omega^n + \Delta t\, \dot{\omega}^n$.
16: Likewise for the body acceleration at time step $n+1$: $\dot{v}^\star := v^n + \Delta t\, \ddot{v}^{n-1/2}$ and $\dot{\omega}^\star := \dot{\omega}^n + \Delta t\, \ddot{\omega}^{n-1/2}$.
17: Calculate a predictor $q^\star$ by solving (16) using x_c^{n+1}, $\omega^\star$, $\dot{v}^\star$, and $\dot{\omega}^\star$.
18: Form a second-order predictor of the gauge variable at time step $n+1$: $\varphi^\star := 2\varphi^n - \varphi^{n-1}$.
19: Define $\mathbf{u}_b^\star := \mathbf{u}_b^\star(x) = v^\star + \omega^\star \times (x - x_c^{n+1})$.
20: Calculate $\mathbf{m}^\star$ such that

$$\begin{cases} \rho\big(\frac{1}{\Delta t}(\mathbf{m}^\star - \mathbf{m}^n) + \mathbf{a_u}\big) = -\nabla q^\star + \mu \Delta \mathbf{m}^\star & \text{in } \Omega_f \\ \mathbf{m}^\star \cdot \mathbf{n} = \mathbf{u}_b^\star \cdot \mathbf{n} \text{ and } \mathbf{m}^\star \cdot \boldsymbol{\tau} = \mathbf{u}_b^\star \cdot \boldsymbol{\tau} + \nabla\varphi^\star \cdot \boldsymbol{\tau} & \text{on } \Gamma \\ \mathbf{m}^\star \cdot \mathbf{n} = \mathbf{u}_\partial^{n+1} \cdot \mathbf{n} \text{ and } \mathbf{m}^\star \cdot \boldsymbol{\tau} = \mathbf{u}_\partial^{n+1} \cdot \boldsymbol{\tau} + \nabla\varphi^\star \cdot \boldsymbol{\tau} & \text{on } \partial\Omega_f \setminus \Gamma. \end{cases}$$

21: Project $\mathbf{m}^\star$ to find $\mathbf{u}^\star = \mathbb{P}(\mathbf{m}^\star)$.
Corrector step
22: Define $\mathbf{a}_\mathbf{u}^\star := \nabla \cdot (\mathbf{u}^\star \mathbf{u}^\star)$.
23: Calculate $\mathbf{m}^{n+1}$ such that

$$\begin{cases} \rho\big(\frac{1}{\Delta t}(\mathbf{m}^{n+1} - \mathbf{m}^n) + \frac{1}{2}(\mathbf{a_u} + \mathbf{a}_\mathbf{u}^\star)\big) = -\frac{1}{2}(\mathbf{f}_q + \nabla q^\star) + \frac{1}{2}\big[\mathbf{f_m} + \mu\Delta\mathbf{m}^{n+1}\big] & \text{in } \Omega_f \\ \mathbf{m}^{n+1} \cdot \mathbf{n} = \mathbf{u}_b^\star \cdot \mathbf{n} \text{ and } \mathbf{m}^{n+1} \cdot \boldsymbol{\tau} = \mathbf{u}_b^\star \cdot \boldsymbol{\tau} + \nabla\varphi^\star \cdot \boldsymbol{\tau} & \text{on } \Gamma \\ \mathbf{m}^{n+1} \cdot \mathbf{n} = \mathbf{u}_\partial^{n+1} \cdot \mathbf{n} \text{ and } \mathbf{m}^{n+1} \cdot \boldsymbol{\tau} = \mathbf{u}_\partial^{n+1} \cdot \boldsymbol{\tau} + \nabla\varphi^\star \cdot \boldsymbol{\tau} & \text{on } \partial\Omega_f \setminus \Gamma. \end{cases}$$

24: Project $\mathbf{m}^{n+1}$ to find $\mathbf{u}^{n+1} = \mathbb{P}(\mathbf{m}^{n+1})$ and $\nabla\varphi^{n+1} = \mathbf{m}^{n+1} - \mathbf{u}^{n+1}$.
25: Compute $\varphi_t^{n+1/2} := \frac{1}{\Delta t}(\varphi^{n+1} - \varphi^n)$.
26: Compute $p^{n+1} = q^\star + \frac{\rho}{2}(3\varphi_t^{n+1/2} - \varphi_t^{n-1/2}) - \mu\nabla \cdot \mathbf{m}^{n+1} + \rho G$.
27: Calculate body force and torque using $\mathbf{u}^{n+1}$, p^{n+1}, and $\omega^\star$, thereby determining $\dot{v}^{n+1}$ and $\dot{\omega}^{n+1}$; see text.
28: Update $v^{n+1} = v^n + \frac{1}{2}\Delta t(\dot{v}^n + \dot{v}^{n+1})$ and $\omega^{n+1} = \omega^n + \frac{1}{2}\Delta t(\dot{\omega}^n + \dot{\omega}^{n+1})$.
29: Compute $\ddot{v}^{n+1/2} = \frac{1}{\Delta t}(\dot{v}^{n+1} - \dot{v}^n)$ and $\ddot{\omega}^{n+1/2} = \frac{1}{\Delta t}(\dot{\omega}^{n+1} - \dot{\omega}^n)$.
30: Compute q^{n+1} according to (16) using x_c^{n+1}, ω^{n+1}, $\dot{v}^{n+1}$, and $\dot{\omega}^{n+1}$.
31: If the remeshing on line 13 signalled a topology update, remesh without preserving the topology, transfer all quantities to the new mesh, and adopt its differential operators.

$$\frac{d}{dt}I_b = (R\tilde{I}_bR^\intercal)' = R'\tilde{I}_bR^\intercal + R\tilde{I}_b(R')^\intercal = S(\omega)R\tilde{I}_bR^\intercal - R\tilde{I}_bR^\intercal S(\omega) = S(\omega)I_b - I_bS(\omega).$$

Thus, if $\frac{d}{dt}(I_b\omega) = \tau$, then $I_b\omega' = \tau - I_b'\omega = \tau - (S(\omega)I_b - I_bS(\omega))\omega$. Since $S(\omega)\omega = 0$, we surmise that $\omega' = I_b^{-1}\tau - I_b^{-1}S(\omega)I_b\omega$.

Consider now the design of a predictor–corrector scheme for the configuration of the body, i.e., x_c, v, ω, and R. Because the net force and torque on the body are known at the beginning of each time step, we can compute the body's updated position and orientation at the end of each time step, to second order accuracy (i.e., locally to third order accuracy), without the need for a predictor stage. In particular, for its centre of mass, update

$$x_c^{n+1} = x_c^n + \Delta t\big(v^n + \tfrac{1}{2}\Delta t\, \dot{v}^n\big),$$

where $\dot{v}^n$ is the computed linear acceleration of the body at time step n. Meanwhile, to update the body's orientation, it is convenient to preserve the orthogonality of R; to achieve this, we compute R^{n+1} using an exponential integrator by solving $R' = S(\omega^{n+1/2})R$, where $\omega^{n+1/2} = \omega^n + \frac{1}{2}\Delta t\dot{\omega}^n$ is a second order approximation to ω at time step $n + \frac{1}{2}$. The solution is given by

$$R^{n+1} = \exp\big(\Delta t\, S(\omega^{n+1/2})\big)R^n = \exp\big(S(\Delta t\, \omega^{n+1/2})\big)R^n.$$

The matrix exponential of $S(\alpha)$ can be computed with the aid of Rodrigues' rotation formula,

$$\exp(S(\alpha)) = \exp\big(|\alpha|S(\alpha/|\alpha|)\big) = \mathbb{I} + (\sin|\alpha|)S(\alpha/|\alpha|) + (1 - \cos|\alpha|)S(\alpha/|\alpha|)^2.$$

To update v and ω, which require computing the fluid stress on the body, we employ a predictor–corrector scheme: a predictor stage is used to advance $\mathbf{m}$, thereby determining the fluid stress $\boldsymbol{\sigma}$ at the end of the time step; the predictor of the stress is then used to advance v^{n+1} and ω^{n+1}, similar to the explicit trapezoid rule used for the advection term $\mathbf{u}\cdot\nabla\mathbf{u}$ seen in previous applications. One aspect of the predictor–corrector scheme, however, deserves special treatment. Recall that, within the time-stepping scheme for the two-phase interfacial gauge methods, the surface tension of the predictor interface is used to form a predictor for $\mathbf{m}^\star$: experiments indicate that doing so improves the stability of the time stepping scheme. Likewise, in this application, it is advantageous to use a predictor of q within the calculation for determining the predictor $\mathbf{m}^\star$. However, we note the following: in order to calculate q, according to (13)–(15), we need to have knowledge of the force and torque on the body at the end of the time step; this, in turn, requires the integral of the stress over the body boundary, which in turn requires knowledge of the fluid pressure p, which depends on q. Thus there is an apparent coupling, as the boundary condition on q depends on q itself. Nevertheless, just as the boundary condition on $\mathbf{m}$ couples to $\nabla\varphi$ and can be straightforwardly treated with a predictor, the same can be done to weakly decouple the calculation of the force and torque on the body from the calculation of $\mathbf{m}$. This is achieved by forming a predictor $q^\star$ whose boundary condition is determined by a second order predictor of the net force and torque on the body, by keeping track of $\dot v$ and $\dot\omega$, akin to keeping track of φ_t within the interfacial gauge methods of §2.2. The overall time stepping strategy is detailed in Algorithm 4, with the result that the coupling of the rigid body to the fluid is handled in such a way that is stable and high-order accurate in both space and time. With regard to its implementation, a few remarks are in order:

- The projection operator $\mathbb{P}$ employed by Algorithm 4 is defined as follows:

$$\text{Given } \mathbf{m}:\Omega\to\mathbb{R}^d,\ \mathbb{P}(\mathbf{m})=\mathbf{m}-\nabla\psi \text{ where } \begin{cases} \Delta\psi=\nabla\cdot\mathbf{m} & \text{in } \Omega\\ \partial_n\psi=\mathbf{m}\cdot\mathbf{n}-\mathbf{u}_b\cdot\mathbf{n} & \text{on } \Gamma\\ \partial_n\psi=\mathbf{m}\cdot\mathbf{n}-\mathbf{u}_\partial\cdot\mathbf{n} & \text{on } \partial\Omega\setminus\Gamma. \end{cases}$$

  Its implementation is similar to that discussed in §2.3.7 of part one [1].
- Given values for x_c, ω, $\dot v$, and $\dot\omega$, the scalar q is determined by solving

$$\begin{cases} \Delta q=0 & \text{in } \Omega_f\\ \partial_n q=-\rho\big(\dot v+\dot\omega\times(x-x_c)+\omega\times(\omega\times(x-x_c))\big)\cdot\mathbf{n} & \text{on } \Gamma\\ \partial_n q=0 & \text{on } \partial\Omega_f\setminus\Gamma. \end{cases} \tag{16}$$

- On line 27 of Algorithm 4, forces on the body are computed (assuming a Cartesian coordinate system) as follows: (i) calculate $\sigma_{ij}=-p\delta_{ij}+\mu(G_iu_j+G_ju_i)$, where G is the discrete gradient operator defined in §2.3 of part one [1], taking into account the lifting data arising from $\mathbf{u}_b$ on Γ; (ii) compute the integrals $f=\int_\Gamma\boldsymbol{\sigma}\mathbf{n}+mg$ and $\tau=\int_\Gamma(x-x_c)\times\boldsymbol{\sigma}\mathbf{n}$ using the quadrature nodes and weights determined by the construction of the current (active) implicit mesh; and, last, (iii) outputting $\dot v=f/m$ and $\dot\omega=R\tilde I_b^{-1}R^\intercal(\tau-S(\omega)R\tilde I_bR^\intercal\omega)$.
- For the second and subsequent time steps, the calculation of q is decoupled from the calculation of body force and torque by using a predictor of the quantities $\dot v$ and $\dot\omega$. However, at time $t=0$, these quantities are a priori unknown (and are in general nonzero), and so the body forces must be computed simultaneously with the initial value of q at $t=0$. The coupling—wherein q is a harmonic function with boundary conditions that depend on surface integrals of itself (as well as ω and $\mathbf{u}$ at $t=0$ and gravity)—is a linear problem for q. It may be possible to solve for q using techniques involving Neumann-to-Dirichlet maps, however, instead, we solve for q using a simple fixed-point iterative method with damping, as follows:
  - Initialise $q^{[0]}\equiv 0$.
  - For $k=0,1,2,\dots$ until convergence:
    - Calculate the forces on the body by using $q^{[k]}$ to output $\dot v^{[k+1]}$ and $\dot\omega^{[k+1]}$.
    - Use $\dot v^{[k+1]}$ and $\dot\omega^{[k+1]}$ to calculate $\tilde q$ according to (16).
    - Define $q^{[k+1]}=\frac12 q^{[k]}+\frac12\tilde q$.
  - Set $q^0=q^{[\infty]}$, $\dot v^0=v^{[\infty]}$, and $\dot\omega^0=\omega^{[\infty]}$.

  Experiments indicated that a small amount of damping is required to ensure convergence of the fixed-point method; the factor of 50% used here is conservative. In practice, approximately 10–20 iterations suffices to initialise q, $\dot v$, and $\dot\omega$ at $t=0$. Furthermore, for the applications considered in this work, which consider a body starting from rest, it suffices to initialise the remaining variables in Algorithm 4 as $\varphi^{-1}=\varphi_t^{-1/2}=0$ and $\ddot v^{-1/2}=\ddot\omega^{-1/2}=0$. Although the true values for these quantities are in general nonzero, initialising them as such reduces the local truncation error to second order, but only for the very first time step; thus the time stepping method remains globally second order.

2.3.3. Convergence tests

To examine the temporal and spatial order of accuracy of the presented rigid body interfacial gauge method, we consider a two-dimensional problem of an ellipsoid of non-uniform density falling under the action of gravity in an enclosed box of fluid. Specifically, the domain is $\Omega=(-\frac12,\frac12)^2$, the fluid has density $\rho=1$ and viscosity $\mu=1$, and gravity is set to $g=-\hat y$. The body is ellipsoidal and in body coordinates is defined by the zero level set of $\phi=\phi(y_1,y_2)=(4y_1)^2+(6y_2)^2-1$ with non-uniform density $\rho=\rho(y_1,y_2)=4(1+e^{12y_1})$. At $t=0$, the centroid of the body is located at the

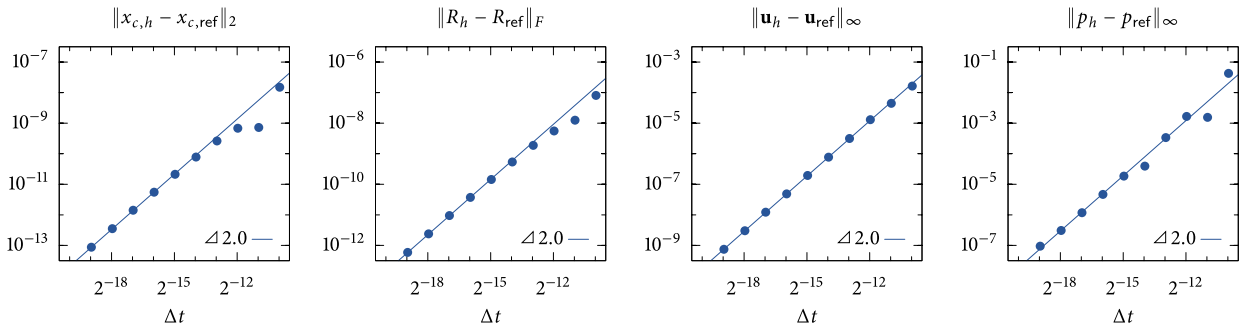

Fig. 15. Results of a temporal convergence analysis for the rigid body interfacial gauge method (13)–(15) implemented by Algorithm 4, showing the maximum-norm error in the computed fluid velocity and pressure as well as the error in the body's computed geometry, i.e., its centre of mass x_c and rotation matrix R, as a function of time step Δt, comparing against the reference solution computed with a time step of $\Delta t \approx 2^{-22}$. Data points represent measured errors at time $t = 0.001$, whereas the lines of indicated slope illustrate the observed order of accuracy.

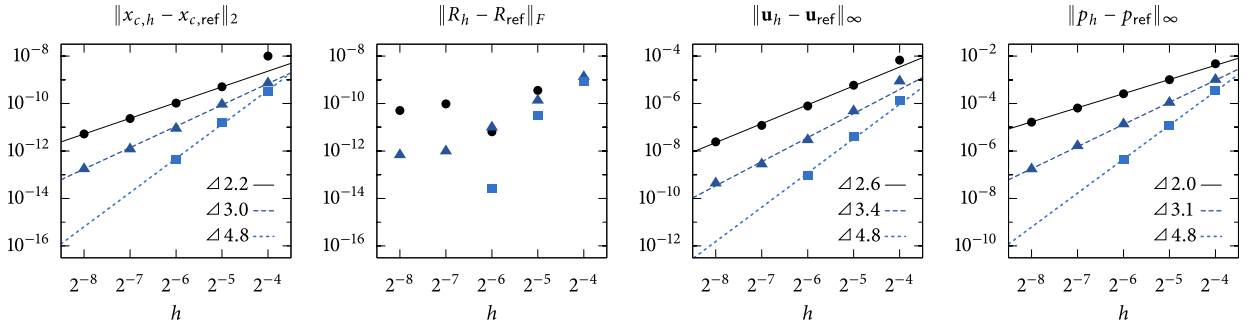

Fig. 16. Results of a spatial order of accuracy convergence analysis for the rigid body interfacial gauge method (13)–(15) showing the maximum-norm error in the computed fluid velocity and pressure as well as the error in the body's computed geometry, i.e., its centre of mass x_c and rotation matrix R, as a function of typical element size h and order p, comparing against a reference solution computed on a 256×256 grid ($h = 2^{-8}$) with $p = 4$ elements. Data points represent measured errors, whereas the lines of indicated slope illustrate the observed order of accuracy. • denotes $p = 2$, ▲ denotes $p = 3$, ■ denotes $p = 4$.

centre of the domain, $(\frac{1}{2}, \frac{1}{2})$, and the body is rotated 22.5° clockwise. It follows that the body's mass is approximately 1.90, it displaces approximately 0.131 fluid mass, and its initial centre of mass is $x_c(t = 0) \approx (0.0951, -0.0394)$. The domain boundary condition is no-slip, $\mathbf{u}_\partial = 0$, and the body is initially stationary, $v|_{t=0} = \omega|_{t=0} = 0$.

Temporal order of accuracy. To study the temporal order of accuracy, a reference solution is computed using a background Cartesian grid of size 32×32 with $p = 4$, i.e., biquartic elements in the implicit mesh dG method, using a time step of $\Delta t \approx 2.44 \times 10^{-7}$. Convergence is examined as a function of the time step Δt by computing the maximum-norm error in fluid velocity and pressure at time $t = 0.001$; at the same instant, we also compute, relative to the reference solution, the error in the body's centre of mass x_c as well as the Frobenius norm of the error in its rotation matrix R. Fig. 15 illustrates the results, and shows that asymptotically second-order temporal accuracy is attained in all quantities.

Spatial order of accuracy. The spatial order of accuracy is examined with a grid convergence study. We consider implicit meshes generated by a uniform background Cartesian grid with sizes ranging from 16×16 to 256×256, together with element orders $p = 2, 3, 4$, corresponding to biquadratic, bicubic, and biquartic elements. A reference solution is computed on the finest mesh with the highest-order elements and a fixed time step of $\Delta t = 1.25 \times 10^{-7}$. Comparing against the reference solution, the error in fluid pressure and velocity in the maximum-norm is measured at time $t = 0.001$, as well as the error in body geometry (centre of mass and rotation matrix). The results are summarised in Fig. 16 and show that the fluid velocity, pressure, and centre of mass of the body are each computed with at least pth-order accuracy. Although the error in the body's rotation matrix does not exhibit a clear asymptotic convergence rate, the relative error is approximately the same order as that exhibited by the centre of mass; the sporadic convergence rate is attributed to R having a greater dependence on grid alignment effects. Similar to the convergence tests in previous applications of the implicit mesh dG framework, note that the spatial order of accuracy is generally sub-optimal (i.e., order p instead of the optimal $p + 1$). However, this may be expected, due to the reliance on accurate computation of the fluid stress tensor to compute the body's net force and torque. Nevertheless, the results examine convergence in the maximum-norm, and thus show that high-order accuracy is obtained throughout the domain as well as immediately adjacent to the body surface. Fig. 17 provides further insight into the considered test problem, the meshes involved, and the qualitative behaviour of the discrete solution: depicted in the figure is the implicitly defined mesh, based on a background 16×16 Cartesian grid, together with the computed results for $p = 4$ biquartic elements and the associated pointwise error.

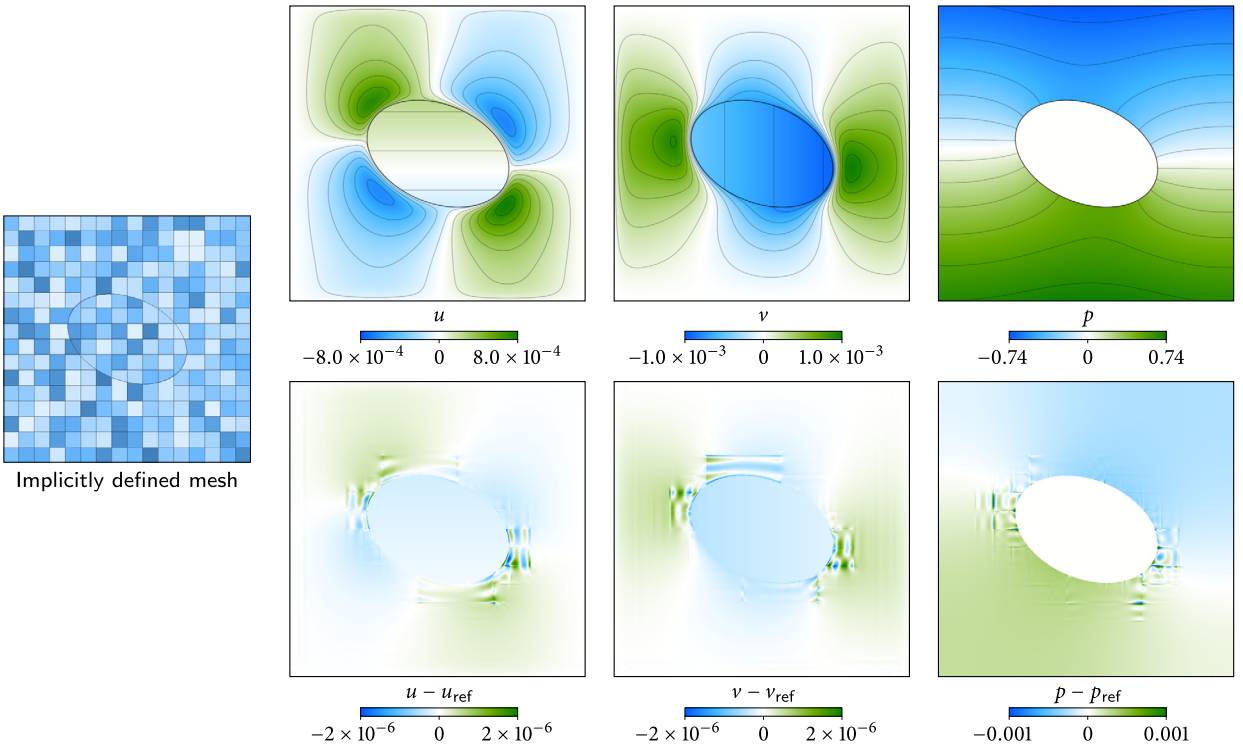

Fig. 17. Corresponding to the spatial order of accuracy grid convergence study of §2.3.3, in the specific case of a 16×16 background Cartesian grid and $p = 4$ biquartic elements, the discrete solution is shown at time $t = 0.001$. The top row plots the computed velocity field $\mathbf{u} = (u, v)$ and pressure field p, which are each piecewise polynomial functions on the depicted implicitly defined mesh; the bottom row plots the error in this discrete solution, which exhibits discontinuities in alignment with the element boundaries. Within the interior of the body, the pressure is left undefined, whereas the velocity (and its associated error) is that of $\mathbf{u}_b = v + \omega \times (x - x_c)$.

2.3.4. A tumbling rigid body

To demonstrate the rigid body interfacial gauge method applied to unsteady flow, we consider here an example of an ellipsoidal body of non-uniform density falling under the action of gravity in a tall column of liquid periodic in the vertical direction. In two dimensions, the chosen domain is $\Omega = (0, 1) \times (0, 4)$ and no-slip boundary conditions are imposed on the left and right sides; the fluid has density $\rho = 1$ and viscosity $\mu = 0.001$; gravity is set to $g = -0.1\,\hat{y}$; and the body is an ellipse with semi-major axis 0.26, semi-minor axis 0.17, and initially rotated 11.25° clockwise, with a non-uniform density such that it is starts top-heavy (i.e., with a centre of mass initially above its centre of buoyancy) with a mass of ≈1.26 displacing ≈0.139 of fluid mass. Since the body is approximately nine times heavier than the fluid it displaces, it falls and accelerates relative to the motion of the fluid. Fig. 18 illustrates the resulting dynamics for this two-dimensional example, computed with a 128×512 background Cartesian grid in the implicit mesh dG framework and $p = 3$ bicubic elements. (It is noted that for this example, the associated mesh is in actuality unnecessarily highly-resolved; in other words, one could use a coarser mesh and attain similar results.) The figures plot the fluid speed and vorticity in a reference frame that follows the body's centre of mass (marked with a dot); the indicated Δy values specify the distance the body has fallen as a percentage of the channel height. Starting from a stationary state at $t = 0$ (i.e., $\mathbf{u} \equiv 0$ and $v = \omega = 0$), the body starts to fall, but viscous effects initially resist the body's inclination to rotate. These effects are shortly overcome owing to the increasing speed: soon after nearly touching the right side ($t = 10$ in Fig. 18) the body tumbles over, and subsequently oscillates from side to side. Due to the periodic boundary conditions in the vertical direction, currents present in the liquid, created by vortex shedding behind the body, return to impact its leading edge, giving rise to further unsteady dynamics. This can be seen in the graphs of Fig. 18, which plot the unsteady forces acting on the body. Using the approximate maximum fluid speed, $U \approx 1.8$, and a length scale equal to the width of the domain, $L = 1$, the Reynolds number $\mathsf{Re} = \rho U L/\mu$ for this flow over the time scale considered is approximately 1800, and the Froude number $\mathsf{Fr} = U/\sqrt{gL}$ is approximately 6.

An analogous example in three dimensions, repeated from [8] but included here for reference, is shown in Fig. 19. The example consists of a three-dimensional domain $\Omega = (0, 1) \times (0, 4) \times (0, 1)$, with no-slip boundary conditions on all sides, except for periodic boundary conditions in the vertical direction $\hat{y}$. The ellipsoid has two semi-major axes 0.26 and semi-minor axis 0.17, with an analogous non-uniform density such that it is initially top-heavy; its mass is ≈0.346 and displaces ≈0.0481 fluid mass. All other parameters are identical to the two-dimensional example considered above. In this example, the results are computed with $p = 3$ tricubic elements in the implicit mesh dG framework, based on a coarser background octree with a mild amount of adaptive mesh refinement: far away from the body, the element size coincides

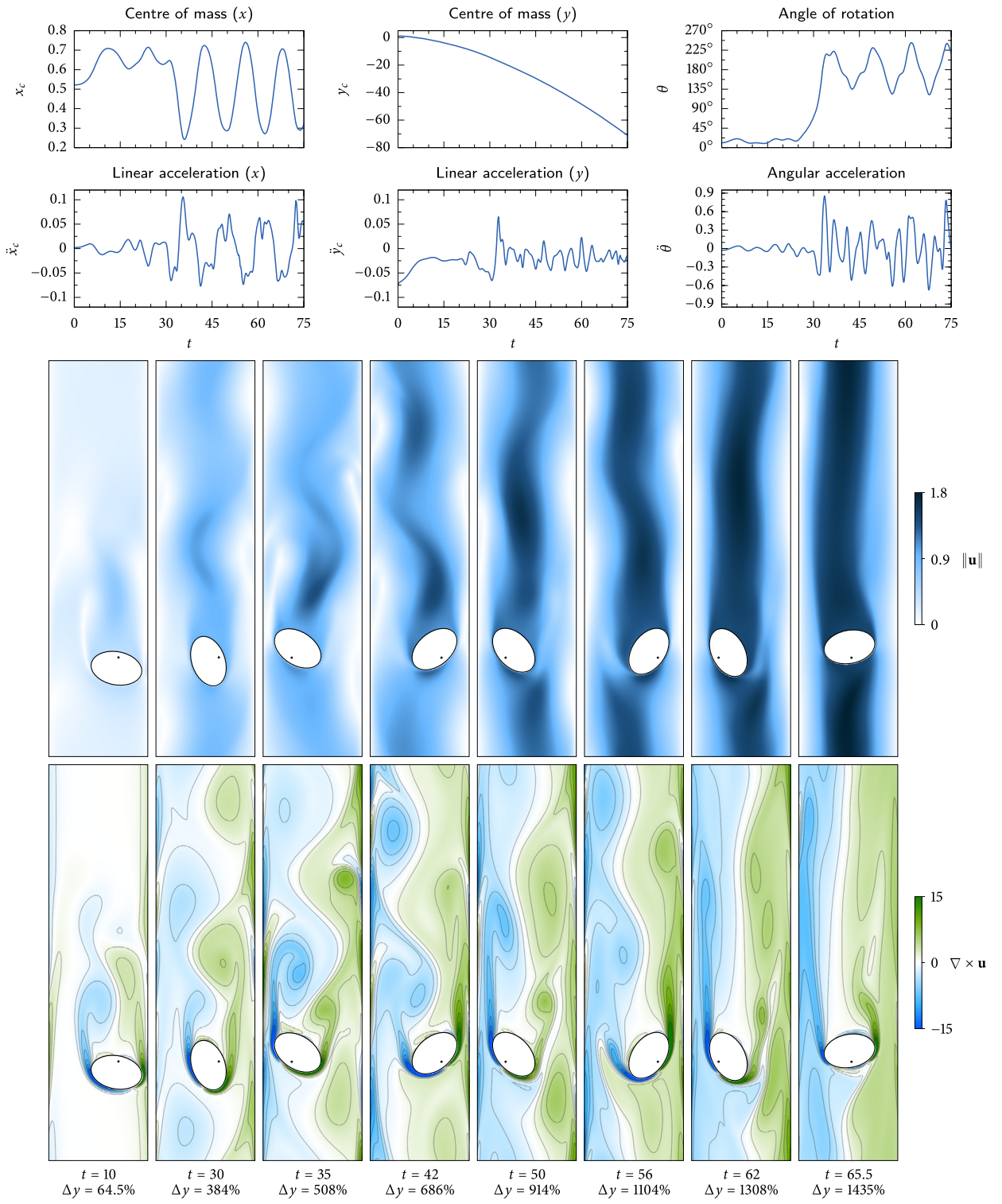

Fig. 18. A tumbling rigid body submersed in a tall channel of liquid falling under the action of gravity. (*top panel*) Graphs of the body's centre of mass x_c, angle of rotation θ, and forces and torques acting on the body (proportional to linear acceleration $\ddot{x}_c$ and angular acceleration $\ddot{\theta} = \dot{\omega}$), showing that the body continues to accelerate in speed in the vertical direction, and, after tumbling over at $t \approx 30$, wobbling side to side. (*bottom panel*) Plots of the fluid speed $\|\mathbf{u}\|$ and vorticity $\omega = \nabla \times \mathbf{u}$ at a sample of points in time. The plots are shown in a reference frame attached to the body's centre of mass; the Δy values indicate the distance the body has fallen as a percentage of the channel height; and the dot within the ellipse indicates the body's centre of mass, which is offset from its centre of buoyancy.

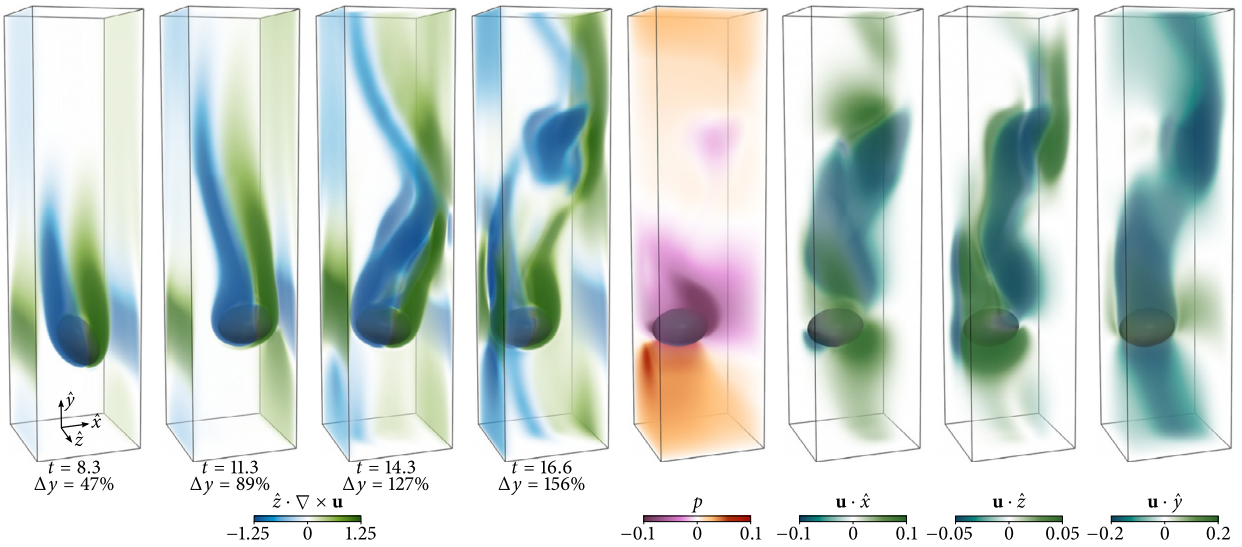

Fig. 19. A tumbling rigid body submersed in a tall channel of liquid falling under the action of gravity (adapted from [8]). (*left four*) Volume renderings of the outward-pointing component of vorticity; from left to right: (i) body rotating clockwise, shortly before flipping over, (ii) shortly after flipping over, (iii) after it has rebounded and unsteadily moved to the other side, and (iv) nearly touching the left wall. The images are shown in a frame of reference attached to the body's centre of mass; the Δy values indicate how far the body has fallen as a percentage of the channel height. (*right four*) Volume renderings of other state quantities at $t = 16.6$ showing pressure p and the components of the velocity field.

with a $8 \times 32 \times 8$ grid; within 25% of the channel height on either side of the body, the elements are twice as small; on the finest level, within 12% of the body, the elements are twice smaller again (corresponding to a $32 \times 128 \times 32$ Cartesian grid). Fig. 19 illustrates some of the computed dynamics, showing qualitatively similar behaviour to the two-dimensional example: as the body falls, viscous effects initially resists the body's inclination to rotate, but are shortly overcome as the body nearly touches the channel boundary, tumbles over and then wobbles side to side.

2.4. Free surface flow

In our last application of the implicit mesh dG framework and of interfacial gauge methods, we consider a different kind of evolving-domain fluid flow problem: free surface flow. In this case, the physics taking place exterior to the fluid are taken as negligible, and so there are no Dirichlet boundary conditions for the fluid velocity on the free surface; instead, a boundary condition is imposed for the normal component of the fluid stress. In particular, we consider here free surfaces in which forces of surface tension balance that of the normal stress of the fluid.

2.4.1. Equations of motion

In the presence of gravity, such that $G = G(x) = g \cdot x$ is the gravitational potential, the velocity field $\mathbf{u}$ of the fluid satisfies the free surface Navier–Stokes equations,

$$\begin{cases} \rho(\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \mu \Delta \mathbf{u} + \rho \nabla G & \text{in } \Omega \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega \\ \mathbf{u} = \mathbf{u}_\partial & \text{on } \partial\Omega \setminus \Gamma \\ \boldsymbol{\sigma}\mathbf{n} = -\gamma\kappa\mathbf{n} & \text{on } \Gamma, \end{cases} \tag{17}$$

where $\mathbf{n}$ on Γ points exterior to the fluid. Here, the domain Ω, the inflow/outflow parts of the domain boundary $\partial\Omega \setminus \Gamma$, and the free surface, Γ, all evolve in time.

In a departure from the interfacial gauge methods considered previously, wherein a Neumann boundary condition for φ is imposed on Γ, in this application we impose Dirichlet boundary conditions for φ on Γ. This relates to the fact that $\mathbf{u} \cdot \mathbf{n}$ on Γ is a priori unknown as well as to the design of stable, idempotent projection operators within an interfacial gauge method, as discussed further in [8]. The interfacial gauge method developed therein is as follows: let $\mathbf{m} : \Omega \to \mathbb{R}^d$, $\varphi : \Omega \to \mathbb{R}$, and $q : \Omega \to \mathbb{R}$ solve

$$\left.\begin{array}{c} \rho(\mathbf{m}_t + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla q + \mu \nabla \cdot (\nabla \mathbf{m} + \nabla \mathbf{m}^\intercal) \\ \mathbf{u} = \mathbf{m} - \nabla \varphi \\ \Delta \varphi = \nabla \cdot \mathbf{m} \\ \Delta q = 0 \end{array}\right\} \text{ in } \Omega, \tag{18}$$

subject to the free surface interfacial conditions

$$\left.\begin{array}{c} \varphi = 0 \\ \mu(\nabla \mathbf{m} + \nabla \mathbf{m}^{\mathsf{T}})\mathbf{n} = 2\mu\big((D^2\varphi)\mathbf{n} - (\Delta\varphi)\mathbf{n}\big) \\ q = \gamma\kappa - \rho G - \rho\varphi_t \end{array}\right\} \text{ on } \Gamma, \tag{19}$$

and domain boundary conditions

$$\left.\begin{array}{c} \mathbf{m}\cdot\mathbf{n} = \mathbf{u}_\partial \cdot \mathbf{n} \\ \mathbf{m}\cdot\boldsymbol{\tau} = \mathbf{u}_\partial \cdot \boldsymbol{\tau} + \nabla\varphi\cdot\boldsymbol{\tau} \\ \partial_n\varphi = \partial_n q = 0 \end{array}\right\} \text{ on } \partial\Omega \setminus \Gamma. \tag{20}$$

Then, with the associated projection operator, one can show that $\mathbf{u} = \mathbb{P}(\mathbf{m})$ solves (17) with pressure identified as $p = q + \rho\varphi_t - 2\mu\nabla\cdot\mathbf{m} + \rho G$. To capture the evolving interface, we make use of a "one-sided" level set method: we define $\phi : \Omega \to \mathbb{R}$ such that the zero level set of ϕ corresponds to Γ and solve the advection equation $\phi_t + \mathbf{u}\cdot\nabla\phi = 0$ within the evolving domain.

2.4.2. Spatial discretisation

A free surface flow problem is essentially a single phase fluid flow problem with a time-dependent domain. In applying the implicit mesh dG framework, there are a variety of possibilities for treating evolving domains. In the following, a background quadtree/octree grid is used which is guaranteed—at least for a sufficient amount of time—to cover the whole domain. The same cell merging procedure can be used and leads to an evolving mesh in much the same way as the fluid–structure application considered in §2.3. In particular, the elemental level set function ϕ is defined only on the elements within the fluid domain; for the purposes of defining the implicit mesh, the elemental level set function is converted to a cellular level set function, defined on the cells of the background quadtree/octree. To accomplish this, essentially the same construction as discussed in §2.6.3 of part one [1] is used: the only modification necessary occurs within the signed distance function computation in §2.6.2, such that only the polynomials of ϕ on the fluid side of the interface are used within Newton's method (cf., the case of two phase flow, wherein the two elemental polynomials on either side of the interface are averaged). The implementation of essentially every discrete differential operator discussed in §2.3 of part one [1] remains the same: for the purposes of imposing Dirichlet and Neumann boundary conditions, one only needs to reidentify Γ (which in this section denotes the evolving surface) as a true domain boundary. The exception relates to the treatment of the advection terms $\nabla\cdot(\mathbf{u}\phi)$ and $\nabla\cdot(\mathbf{u}\mathbf{u})$ in §2.5, where a numerical flux needs to be specified on Γ. For the level set advection term, the numerical flux is set to zero on Γ since ϕ is zero there. For the advection term for velocity, boundary values for $\mathbf{u}\cdot\mathbf{n}$ on Γ are a priori unknown; in this case, the numerical flux is set to $\mathbf{u}|_\Gamma$.

The projection operator for the interfacial gauge method (18)–(20) is as follows:

$$\text{Given } \mathbf{m} : \Omega \to \mathbb{R}^d,\ \mathbb{P}(\mathbf{m}) = \mathbf{m} - \nabla\psi \text{ where } \left\{\begin{array}{cc} \Delta\psi = \nabla\cdot\mathbf{m} & \text{in } \Omega \\ \psi = 0 & \text{on } \Gamma \\ \partial_n\psi = \mathbf{m}\cdot\mathbf{n} - \mathbf{u}_\partial\cdot\mathbf{n} & \text{on } \partial\Omega\setminus\Gamma. \end{array}\right.$$

Its implementation in terms of the LDG operator $G : V_h \to V_h^d$ (constructed under the assumption of Dirichlet boundary conditions on Γ and Neumann boundary conditions on $\partial\Omega\setminus\Gamma$) is as follows: (i) solve for $\psi \in V_h$ such that

$$\Big[\Big(\sum_{1\le k\le d} G_k^{\mathsf{T}} M G_k\Big) + \tau_0 A_0 + \tau_D A_D\Big]\psi = \sum_{1\le k\le d} G_k^{\mathsf{T}} M m_k - M J(\mathbf{u}_\partial\cdot\mathbf{n}),$$

where $m_i \in V_h$, $1 \le i \le d$, are the components of $\mathbf{m}$, and $J(\mathbf{u}_\partial\cdot\mathbf{n}) \in V_h$ is the lifting of the boundary data such that $\int_\Omega J(\mathbf{u}_\partial\cdot\mathbf{n})v = \int_{\partial\Omega\setminus\Gamma} v^-\mathbf{u}_\partial\cdot\mathbf{n}$ for all $v \in V_h$; and then (ii) compute $\mathbf{u} = \mathbf{m} - G\psi$. Thus, although this projection operator has a mixture of Dirichlet boundary conditions (on Γ) and Neumann boundary conditions (on $\partial\Omega\setminus\Gamma$), the discrete formulation is similar to the projection operators discussed in §2.3.7 of part one [1]. Similarly, putting aside the influence of the LDG penalty parameters, the discrete projection operator is idempotent, which is important in designing accurate and stable interfacial gauge methods [8]. In this case, we may also have a penalty parameter τ_D associated with the Dirichlet part of the domain boundary (i.e., Γ) for the elliptic problem. In practice, it is set to $\tau_D = 1000$; this value is chosen having in mind the performance of the multigrid preconditioned conjugate gradient algorithm: a larger value of τ_D allows the relaxation method to more efficiently impose Dirichlet boundary conditions.

2.4.3. Temporal discretisation

The design of a predictor–corrector scheme for the above interfacial gauge method for free surface flow is similar to that for two-phase flow driven by surface tension: a predictor step is used to form a prediction of the surface tension on the free surface at the next time step, which is then used as part of a corrector step to achieve second-order accuracy in time. The scheme is detailed in Algorithm 5; two brief remarks:

Algorithm 5 Predictor–corrector scheme for the interfacial gauge method for free surface flow, (18)–(20).

1: Define the initial implicit mesh and adopt its differential operators.
2: Initialise $\mathbf{u}^0$, $\mathbf{m}^0$, φ^0, and ϕ^0 at time $t=0$.
3: Determine an appropriate φ^{-1}, $\varphi_t^{-1/2}$ and $\varphi_t^{-3/2}$.
4: Calculate q^0 by solving (21) using $\kappa(\phi^0)$ and $\varphi_t = \frac{3}{2}\varphi_t^{-1/2} - \varphi_t^{-3/2}$.
5: **for** $n = 0, 1, 2, \ldots$ **do**
6: Form a second-order predictor of the gauge variable at time step $n+1$: $\varphi^\star := 2\varphi^n - \varphi^{n-1}$.
7: Form a second-order predictor of $\partial_t\varphi$ at time step $n+1$: $\varphi_t^\star := \frac{5}{2}\varphi_t^{n-1/2} - \frac{3}{2}\varphi_t^{n-3/2}$.
8: Define $a_\phi := \nabla\cdot(\mathbf{u}^n\phi^n)$ and $\mathbf{a_u} := \nabla\cdot(\mathbf{u}^n\mathbf{u}^n)$.
9: Define $\mathbf{f}_q := \nabla q^n$ and $\mathbf{f_m} := \mu\nabla\cdot\left(\nabla\mathbf{m}^n + \nabla\mathbf{m}^{n\,\mathsf{T}}\right)$.
Predictor step
10: Update the level set function: $\phi^\star = \phi^n - \Delta t\, a_\phi$.
11: Define a predictor mesh—preserving the topology of the current mesh—using $\phi^\star$ and its implicitly defined interface $\Gamma^\star$.
12: Transfer all necessary quantities to the new mesh and adopt its differential operators.
13: Calculate $q^\star$ by solving (21) using $\kappa(\phi^\star)$ and $\varphi_t^\star$.
14: Calculate $\mathbf{m}^\star$ such that

$$\begin{cases} \rho\left(\frac{1}{\Delta t}(\mathbf{m}^\star - \mathbf{m}^n) + \mathbf{a_u}\right) = -\nabla q^\star + \mu\nabla\cdot\left(\nabla\mathbf{m}^\star + \nabla\mathbf{m}^{\star\,\mathsf{T}}\right) & \text{in } \Omega \\ \mu\left(\nabla\mathbf{m}^\star + \nabla\mathbf{m}^{\star\,\mathsf{T}}\right)\cdot\mathbf{n} = 2\mu\left((D^2\varphi^\star)\mathbf{n} - (\Delta\varphi^\star)\mathbf{n}\right) & \text{on } \Gamma \\ \mathbf{m}^\star\cdot\mathbf{n} = \mathbf{u}_\partial^{n+1}\cdot\mathbf{n} \text{ and } \mathbf{m}^\star\cdot\boldsymbol{\tau} = \mathbf{u}_\partial^{n+1}\cdot\boldsymbol{\tau} + \nabla\varphi^\star\cdot\boldsymbol{\tau} & \text{on } \partial\Omega\setminus\Gamma. \end{cases}$$

15: Project $\mathbf{m}^\star$ to find $\mathbf{u}^\star = \mathbb{P}(\mathbf{m}^\star)$.
Corrector step
16: Define $\mathbf{a}_\mathbf{u}^\star := \nabla\cdot(\mathbf{u}^\star\mathbf{u}^\star)$.
17: Update the level set function: $\phi^{n+1} = \phi^n - \frac{1}{2}\Delta t\left(a_\phi + \nabla\cdot(\mathbf{u}^\star\phi^\star)\right)$.
18: Define the new mesh—preserving the topology of the current mesh—using ϕ^{n+1} and its implicitly defined interface Γ^{n+1}.
19: Transfer all necessary quantities to the new mesh and adopt its differential operators.
20: Calculate q^{n+1} by solving (21) using $\kappa(\phi^{n+1})$ and $\varphi_t^\star$.
21: Calculate $\mathbf{m}^{n+1}$ such that

$$\begin{cases} \rho\left(\frac{1}{\Delta t}(\mathbf{m}^{n+1} - \mathbf{m}^n) + \frac{1}{2}(\mathbf{a_u} + \mathbf{a}_\mathbf{u}^\star)\right) = -\frac{1}{2}(\mathbf{f}_q + \nabla q^{n+1}) + \frac{1}{2}\left[\mathbf{f_m} + \mu\nabla\cdot\left(\nabla\mathbf{m}^{n+1} + \nabla\mathbf{m}^{n+1\,\mathsf{T}}\right)\right] & \text{in } \Omega \\ \mu\left(\nabla\mathbf{m}^{n+1} + \nabla\mathbf{m}^{n+1\,\mathsf{T}}\right)\cdot\mathbf{n} = 2\mu\left((D^2\varphi^\star)\mathbf{n} - (\Delta\varphi^\star)\mathbf{n}\right) & \text{on } \Gamma \\ \mathbf{m}^{n+1}\cdot\mathbf{n} = \mathbf{u}_\partial^{n+1}\cdot\mathbf{n} \text{ and } \mathbf{m}^{n+1}\cdot\boldsymbol{\tau} = \mathbf{u}_\partial^{n+1}\cdot\boldsymbol{\tau} + \nabla\varphi^\star\cdot\boldsymbol{\tau} & \text{on } \partial\Omega\setminus\Gamma. \end{cases}$$

22: Project $\mathbf{m}^{n+1}$ to find $\mathbf{u}^{n+1} = \mathbb{P}(\mathbf{m}^{n+1})$ and $\nabla\varphi^{n+1} = \mathbf{m}^{n+1} - \mathbf{u}^{n+1}$.
23: Compute $\varphi_t^{n+1/2} := \frac{1}{\Delta t}(\varphi^{n+1} - \varphi^n)$.
24: Optionally, compute pressure at time step $n+1$ according to: $p^{n+1} := q^{n+1} + \frac{\rho}{2}\left(3\varphi_t^{n+1/2} - \varphi_t^{n-1/2}\right) - 2\mu\nabla\cdot\mathbf{m}^{n+1} + \rho G$.
25: If the remeshing on line 11 signalled a topology update, remesh without preserving the topology, transfer all quantities to the new mesh, and adopt its differential operators.

- To compute q, we solve the elliptic interface problem

$$\begin{cases} \Delta q = 0 & \text{in } \Omega \\ q = \gamma\kappa(\phi) - \rho\varphi_t - \rho G & \text{on } \Gamma \\ \partial_n q = 0 & \text{on } \partial\Omega, \end{cases} \tag{21}$$

  where $\kappa(\phi)$ is the computed (see Appendix C) mean curvature of the level set function ϕ^{n+1} or the predictor $\phi^\star$, and φ_t is computed by a second-order accurate predictor with $\varphi_t^\star := \frac{5}{2}\varphi_t^{n-1/2} - \frac{3}{2}\varphi_t^{n-3/2}$.
- In order to determine the Dirichlet boundary conditions on Γ for $\mathbf{m}^\star$ or $\mathbf{m}^{n+1}$, in the backward Euler or Crank–Nicolson solve, respectively, one must calculate the Hessian of $\varphi^\star$. To accomplish this, the same method as that used by the two-phase interfacial gauge method is used, see §2.2.2.

2.4.4. Convergence tests

With the same motivation as that underlying the design of the grid convergence study in §2.2.3 for two-phase surface tension dynamics, we adopt here a similar analysis. Specifically, we consider a test problem with interface initially a sine-wave of one wavelength (given by the zero level set of $\phi = \phi(x,y) = y - 0.025\cos 2\pi x$) in a channel $-\frac{1}{2} \le x \le \frac{1}{2}$, $y \ge -\frac{1}{2}$, with flow periodic in the x direction and a no-slip boundary condition on the bottom wall $\{y = -\frac{1}{2}\}$. We set $\gamma = 10$ and $g = -10\hat{y}$, which results in surface tension and gravitational forces of similar magnitude; meanwhile, the fluid has density $\rho = 1$ and viscosity $\mu = 1$. Combined, these parameters are such that the dynamics are mildly resolvable on a coarse mesh with $h = 2^{-3}$. As in other convergence tests, we employ a maximum-norm metric on the domain $\Omega\cap\tilde{\Omega}$, where Ω and $\tilde{\Omega}$ are the meshes of two different simulations. To initialise the interfacial gauge method, in the time stepping algorithm of Algorithm 5 we set $\mathbf{m}^0 = \mathbf{u}^0 = 0$ and $\varphi^0 = \varphi_t^{-1/2} = \varphi_t^{-3/2} = 0$.

Temporal order of accuracy. To examine the temporal order of accuracy, a reference solution is computed using a background Cartesian grid with cell size $h = 2^{-5}$, $p = 4$ biquartic elements, and a fixed time step $\Delta t \approx 2.4\times 10^{-6}$. The maximum-norm error in the computed solution at time $t = 0.01$ is then measured for a sequence of solutions computed with progressively larger time steps. The results are shown in Fig. 20 and show that all state quantities asymptotically attain second-order temporal accuracy.

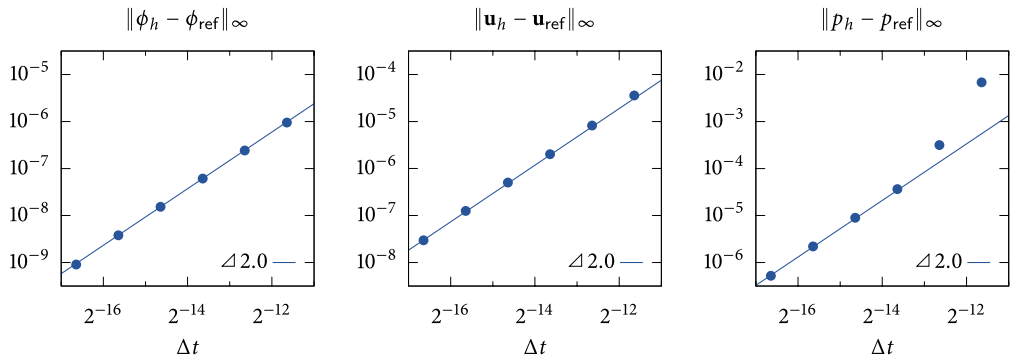

Fig. 20. Results of a temporal convergence analysis for the free surface interfacial gauge method (18)–(20) implemented by Algorithm 5 showing the maximum-norm error in the computed velocity field, pressure, and level set function, as a function of time step Δt, comparing against the reference solution computed with a time step of $\Delta t \approx 2^{-18.6}$. Data points represent measured errors at time $t = 0.01$, whereas the lines of indicated slope illustrate the observed order of accuracy.

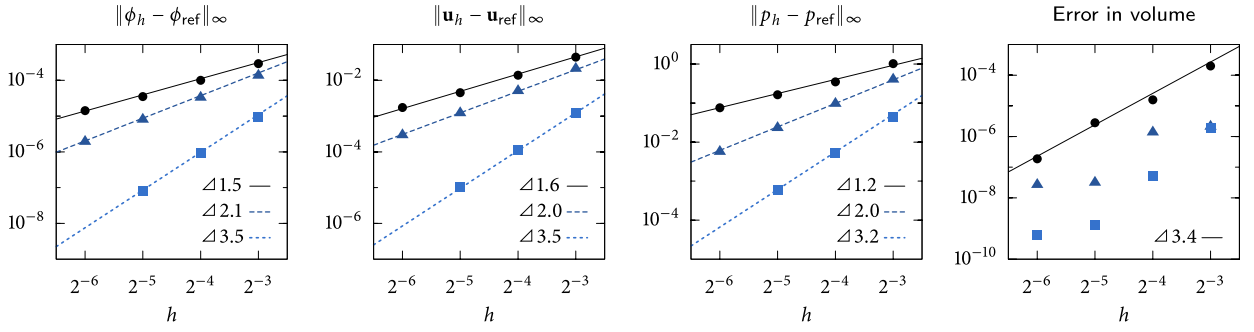

Fig. 21. Results of a spatial order of accuracy convergence analysis for the free surface interfacial gauge method (18)–(20) showing the maximum-norm error in the computed velocity field, pressure, and level set function, as a function of typical element size h and order p, comparing against a reference solution computed using a background Cartesian grid with cell size $h = 2^{-6}$ and $p = 4$ elements. Also shown is the error in volume, measured at time $t = 0.01$. Data points represent measured errors, whereas the lines of indicated slope illustrate the observed order of accuracy. • denotes $p = 2$, ▲ denotes $p = 3$, ■ denotes $p = 4$.

Spatial order of accuracy. To study the spatial order of accuracy, a reference solution is computed on the finest mesh ($h = 2^{-6}$) with $p = 4$ biquartic elements, using a fixed time step of $\Delta t = 10^{-5}$, empirically determined such that temporal errors are negligible compared to spatial errors. With a range of background Cartesian grids of cell sizes 2^{-6} to 2^{-3}, we consider $p = 2, 3$, and 4, corresponding to biquadratic, bicubic and biquartic elements. Comparing against the reference solution, Fig. 21 shows the maximum-norm error for the fluid velocity, pressure, and level set function, and also shows the error in conservation of mass, all at time $t = 0.01$. In this case of free surface flow, we observe slightly reduced convergence rates compared to the observed order of accuracy for the case of two-phase surface tension dynamics. That is, for $p = 2$, the order of accuracy is approximately 1.5; for $p = 3$, it is approximately 2.0; and for $p = 4$, the convergence rate is approximately 3.5. This reduction is attributed to the greater influence the stress tensor boundary condition has on the ensuing dynamics: in other words, the boundary conditions on $\mathbf{m}$ involve the Hessian of φ, which is not computed with optimal order accuracy on the unstructured meshes considered in this work. Nevertheless, these results examine convergence in the maximum norm, and thus the ability to accurately resolve dynamics adjacent to the free surface. Fig. 22 provides further insight into the considered test problem, the mesh involved, and the qualitative behaviour of the discrete solution: depicted in the figure is the implicitly defined mesh, based on a background Cartesian grid of cell size $h = 2^{-3}$, together with the computed results for $p = 4$ biquartic elements and the associated pointwise error.

2.4.5. Flow over a submersed obstacle

Fig. 23 demonstrates the free surface flow framework with a three-dimensional example in which fluid flows over a submersed obstacle. The example consists of a unit square bed with a spherical obstacle of radius 0.125 partially emerging from the bottom; the flow is periodic on all four sides and is forced by an external force $\mathbf{f} = 0.5\hat{x}$ in the downstream direction $\hat{x}$; surface tension is relatively weak, $\gamma = 0.01$, whereas gravity is stronger $g = -0.5\,\hat{y}$; the fluid has density $\rho = 1$ and viscosity $\mu = 0.001$, and at $t = 0$ is initially stationary with a horizontal (flat) surface a height ≈ 0.15 above the bed. Computed with the implicit mesh dG framework using $p = 3$ tricubic elements, on a mesh implicitly defined by the obstacle and a background Cartesian grid $32 \times 4 \times 32$ in size, Fig. 23 shows the flow at time $t = 8$. At this time, the flow has developed eddies behind the obstacle, as shown by the mid-level streamlines, the vorticity generation on the bed

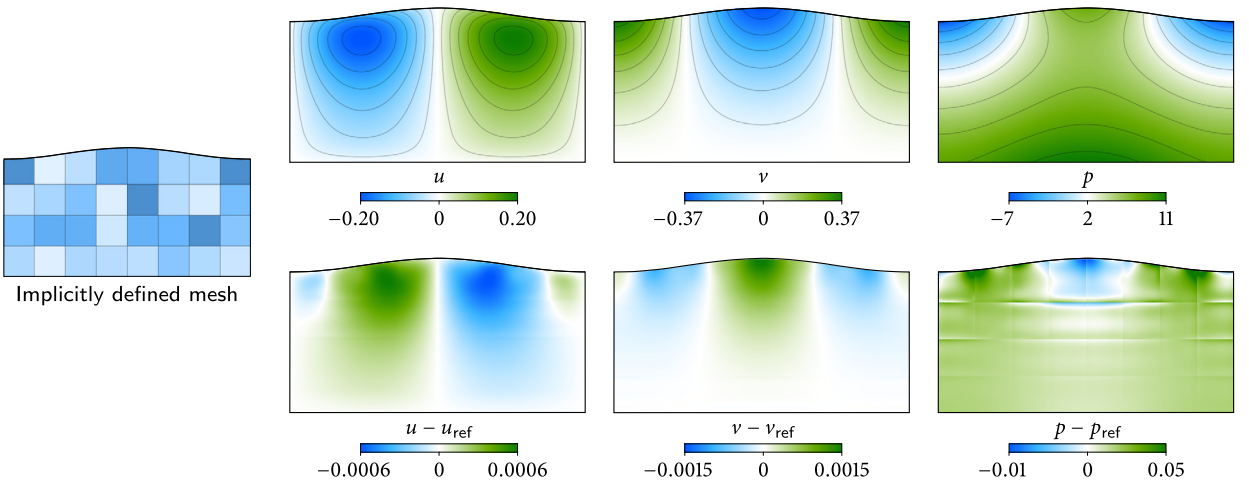

Fig. 22. Corresponding to the spatial order of accuracy grid convergence study of §2.4.4, in the specific case of a relatively coarse background Cartesian grid with cell size $h = 2^{-3}$ and $p = 4$ biquartic elements, the discrete solution is shown at time $t = 0.01$. The top row plots the computed velocity field $\mathbf{u} = (u, v)$ and pressure field p, which are each piecewise polynomial functions on the depicted implicitly defined mesh; the bottom row plots the error in this discrete solution, which exhibits discontinuities in alignment with the element boundaries.

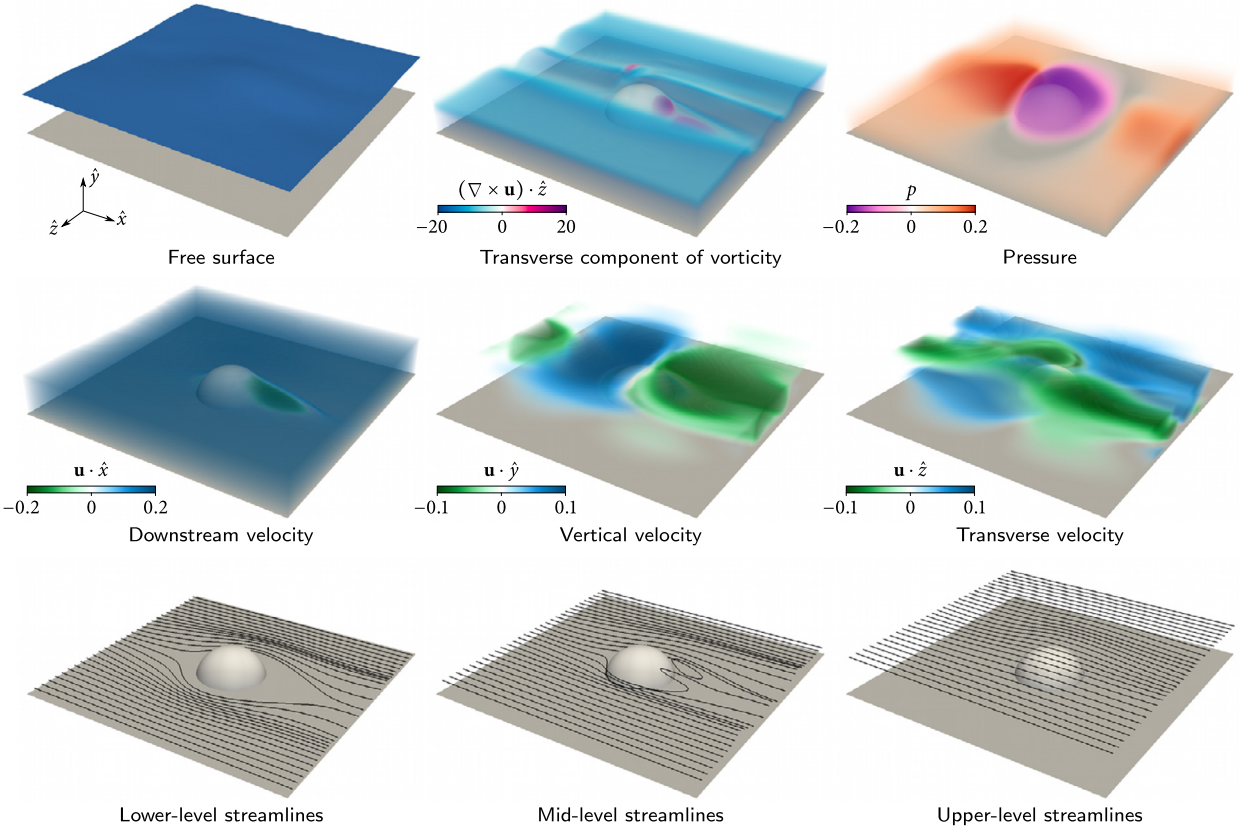

Fig. 23. Free surface flow over a submersed obstacle. (*top left*) Free surface profile, showing slight undulations caused by the influence of the submersed obstacle underneath. (*bottom row*) Streamlines computed by seeding massless particles on the upstream side of the domain at various heights. (*remainder*) Volume renderings of the stated quantities; in each case, the colour scale has been clamped in order to make the visualisation clearer. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

surface/obstacle, and in the region behind the obstacle in which $\mathbf{u} \cdot \hat{x}$ is negative. The upper-level streamlines show the flow above the obstacle is relatively unaffected, however the free surface shape nevertheless develops undulations in its profile due to the influence of the submersed obstacle. With a velocity scale approximately equal to the maximum fluid speed,

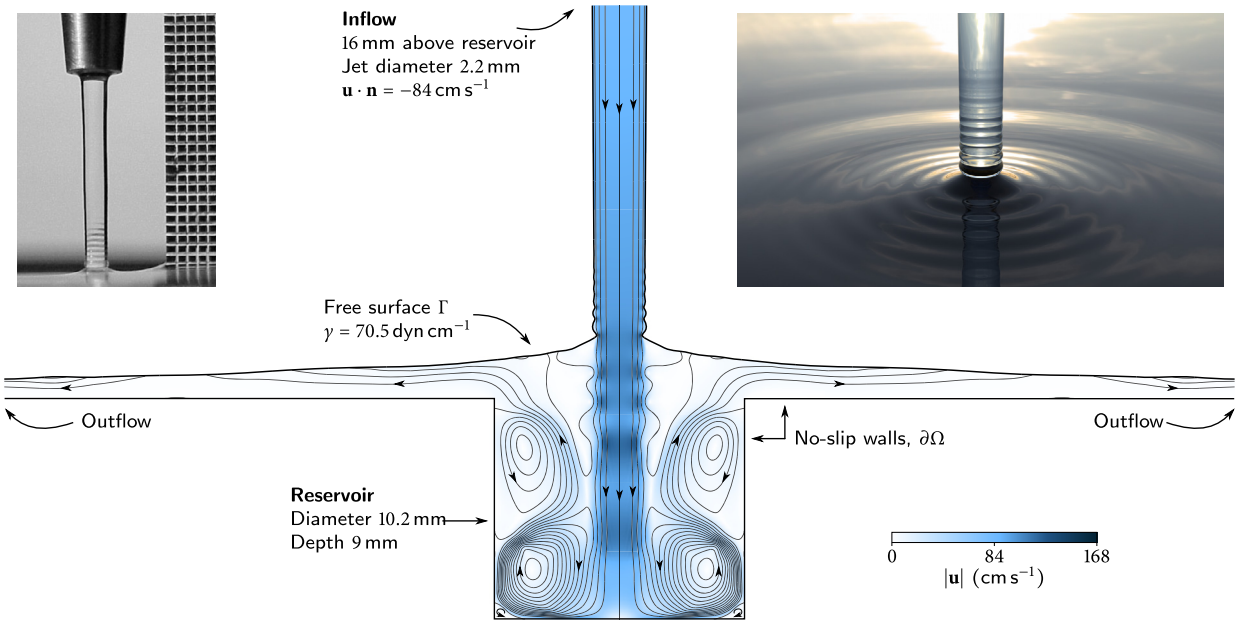

Fig. 24. Water ripples in a free surface flow. (*left inset*) Experimental image reproduced with permission from Hancock and Bush [26] showing a jet of water exiting a nozzle and entering a reservoir. When the stream is steady, surface tension generated capillary waves can be seen travelling up the stream. These ripples are caused by the Plateau–Rayleigh instability, which acts to collapse a cylindrical tube of liquid; however, in this circumstance, the waves are steadily maintained by the constant inflow of water from the jet. The grid seen in the photo is millimetric. (*main*) Results of an axisymmetric simulation, computed with the implicit mesh dG framework and the free surface flow interfacial gauge method (18)–(20), reproducing this ripple phenomenon. Shown are the streamlines of the flow and the speed of the water, as well as the chosen domain layout and boundary conditions: the jet radius, inflow speed, and physical parameters such as water density and viscosity, match those of the experiment—note that the computed results reproduce the ripple phenomenon at the base of the jet, the shape and wavelengths of which are in good agreement with the experimental result. See also Fig. 25 for zoomed-in plots of the computed fluid pressure and vorticity. (*right inset*) A ray-traced rendering of the numerical results, showing capillary waves travelling outwards on the surface of the water.

$U = 3$, and the length scale of the domain, $L = 1$, the Reynolds number $\mathrm{Re} = \rho UL/\mu$ for this flow is approximately 3000, the Froude number $\mathrm{Fr} = U/\sqrt{gL}$ is approximately 4, the Capillary number $\mathrm{Ca} = \mu U/\gamma$ is approximately 0.3, and the Weber number $\mathrm{We} = \rho U^2 L/\gamma$ is approximately 900.

2.4.6. *Water ripples induced by the Plateau–Rayleigh instability*

In the last example of combining the implicit mesh dG framework with that of interfacial gauge methods, we include and expand upon a challenging free surface flow problem first presented in [8]. The phenomena considered can be readily observed at home: when a steady stream of water of diameter 2 to 5 mm exits from a tap and is obstructed downstream by, e.g., a finger, mildly steady ripples are seen in the stream immediately above the obstruction. These ripples are caused by surface tension and the Plateau–Rayleigh instability that acts to collapse a cylinder of liquid; capillary waves are created, which are then sustained by the steady inflow of liquid from the tap, see, e.g., the experimental image in Fig. 24. In the example considered here, we compare against a experimental study of the Plateau–Rayleigh instability conducted by Hancock and Bush [26], which considers a water jet impinging on a reservoir of water. In that work, it is argued that sufficiently far upstream of the ripples, the jet flow can be considered as plug flow, i.e., with uniform velocity across the circular cross section of the jet. This profile is used as an inflow boundary condition, where, based on quoted experimental parameters and photographs [26], the jet speed is estimated to be 84 $\mathrm{cm\,s^{-1}}$, uniform across the jet of diameter 2.2 mm, with an estimated jet height of 16 mm, while the remaining parameters are taken as $\rho = 1\ \mathrm{g\,cm^{-3}}$, $\mu = 0.012\ \mathrm{g\,cm^{-1}\,s^{-1}}$, $\gamma = 70.5\ \mathrm{dyn\,cm^{-1}}$, and $g = 980\ \mathrm{cm\,s^{-2}}$. In the cited experiments, the water jet strikes a large reservoir of water, eventually resulting in a steady flow once the transient unsteady flow decays. Because modelling a large container of water is computationally expensive, the model of [8] considered instead a mildly deep "well" situated underneath a flat table, so as to ensure the outflow boundary conditions are far away from the jet; Fig. 24 shows the chosen domain. Finally, the symmetry of the problem is exploited by computing in cylindrical coordinates, seeking an axisymmetric solution; details on adapting the implicit mesh dG framework to axisymmetric computations are given in §Appendix A.

The small viscosity of water makes for a challenging computation, due in part to the lack of an ideal initial condition. For example, if the jet of water is simply instantaneously turned on, a high degree of turbulent flow is obtained, taking an impractical number of time steps to find a steady state for the ripple profile. As an alternative strategy, in [8], the liquid within the jet and reservoir is initially taken to be 15 times more viscous than water: this allows a steady free surface flow profile to be found more quickly, exhibiting no ripples. After this flow profile is found, the viscosity of the liquid is then slowly ramped down to water's, allowing the ripples to slowly form. Once the final prescribed viscosity is attained,

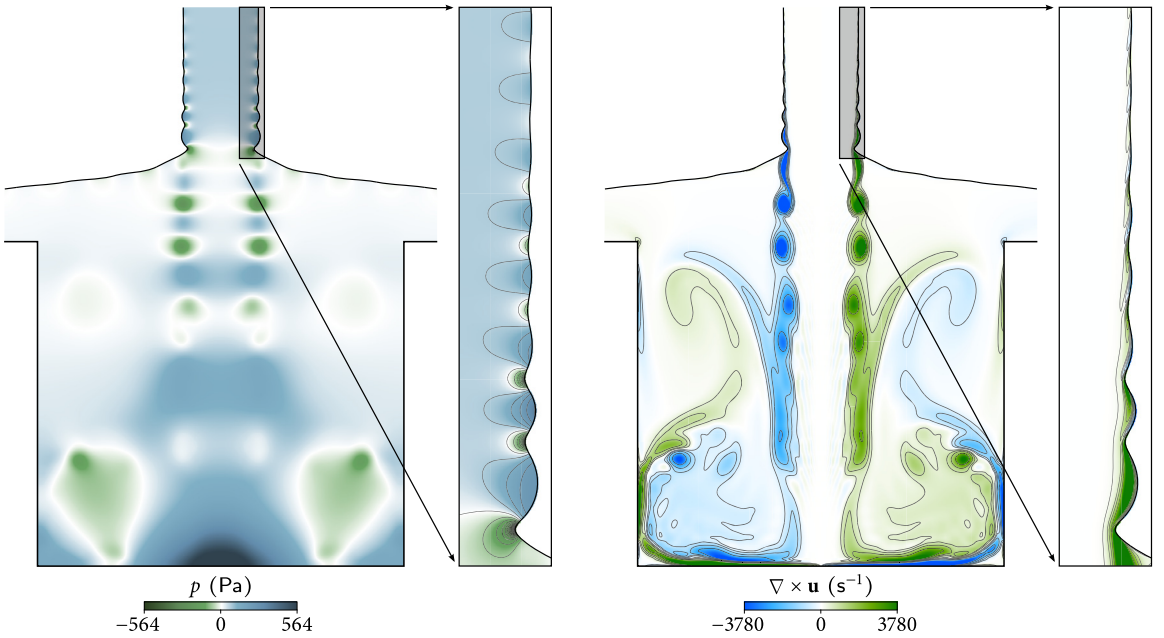

Fig. 25. Fluid pressure and vorticity in the free surface flow of Fig. 24. In both figures, a zoomed-in view of the ripple profile is given, illustrating the wide range of scales taking place.

this mechanism results in a flow in which the free surface is approximately stationary, avoiding the unnecessary onset of turbulence.

Fig. 24 shows the computed flow profile a short time after the steady ripple profile is attained. The results are computed with the axisymmetric implicit mesh dG framework, with $p = 3$ bicubic elements, on an adaptively refined mesh composed of approximately 30,000 elements; the mesh has 5 levels of refinement, ranging from the largest elements within the interior of the well (corresponding to a background 128×128 Cartesian grid covering the domain of size 25 cm $\times$ 25 cm) to elements 16 times smaller surrounding the ripples, where the resolution requirements are highest. Regarding the ripple geometry, as seen in Fig. 24, a good agreement between the experiment and computation is obtained, serving as an additional means of validation for the presented Navier–Stokes solver. Note also the wide breadth of scales involved: Fig. 24 shows that the top part of the jet and the outflow profile are relatively steady, whereas the reservoir exhibits many eddies of different sizes, caused, in part, by the vortices that are shed at the base of the ripples, themselves having smaller-scaled dynamics. This latter phenomena is highlighted in Fig. 25, with a zoomed-in view of the ripple profile, fluid pressure, and vorticity; in addition, Fig. 26 shows a time series of the vortex shedding, illustrating that the axisymmetric vortex tubes are shed with nearly constant frequency. Using the maximum fluid velocity (realised at the base of the ripples, $\approx$170 cm s^{-1}) and the radius of the reservoir, the Reynolds number $\mathsf{Re} = \rho U L/\mu$ for this flow is approximately 7200, the Froude number $\mathsf{Fr} = U/\sqrt{gL}$ is approximately 8, the Capillary number $\mathsf{Ca} = \mu U/\gamma$ is approximately 0.03, and the Weber number $\mathsf{We} = \rho U^2 L/\gamma$ is approximately 210.

3. Concluding remarks

In this two-part paper, a high-order accurate discontinuous Galerkin framework based on implicitly defined meshes has been presented, facilitating precise computation of interfacial fluid flow in evolving geometries. The essential construction is as follows: at each time step, the geometry—domain boundaries, multiphase interfaces, and/or rigid bodies—is defined implicitly on the cells of a background quadtree or octree grid. Cells belonging to individual fluid phases are then classified as small, large, or entire for the purposes of cell merging, which subsequently leads to an implicitly defined mesh with curved elements. The mesh is never explicitly parameterised or geometrically reconstructed; instead, the geometry of the mesh is inferred through the mass matrices and face integration rules for the curved elements, which, in turn, are computed with high-order accurate quadrature schemes. The computed integration rules are then used in various dG discretisations via variational statements. Owing to the property that the implicit meshes are automatically interface-conforming, i.e., interfaces are sharply represented by the mesh, physically prescribed jump conditions can be captured with high-order accuracy. According to a series of grid convergence tests which examine convergence in the maximum norm, it was shown that the presented LDG discretisations—for scalar and vector-valued elliptic interface problems with jumps in ellipticity coefficient—yield optimal-order accuracy. That is, for a degree p piecewise polynomial finite element space, the maximum norm error is $\mathcal{O}(h^{p+1})$, where h is the typical element size. Also derived were projection operators used within incompressible fluid flow solvers; on unstructured meshes, such as the implicitly defined meshes used within this work, experiments indicate that

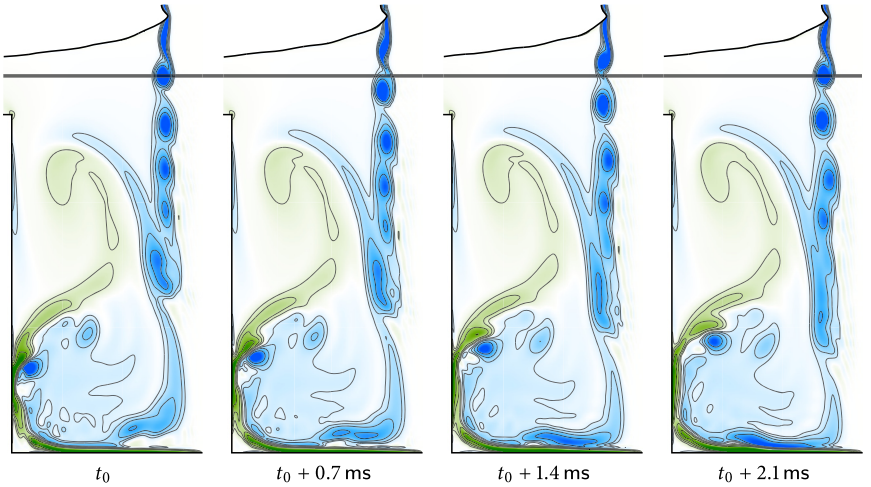

Fig. 26. Vortex shedding in the free surface flow of Fig. 24. Illustrated is a time series of the azimuthal component of the fluid vorticity $\nabla \times \mathbf{u}$ (sharing the same scale as that in Fig. 25) restricted to the left half; overlaid is a horizontal guide line to highlight the downward movement of vortices (vortex tubes by axisymmetry) over a period of 2.1 ms.

the order of accuracy is $p+1$ in computing the scalar gauge variable, and order p in the maximum norm for the computed projected velocity field. Also discussed were various aspects relating to moving interfaces and thus evolving implicitly defined meshes. Transference of state between changing implicit meshes is essentially Eulerian in nature, i.e., polynomials are simply injected into the new mesh and their nodal basis coefficients are copied/preserved. The exception to this property relates to the necessity of creating and destroying elements as the interface travels through the cells of the background quadtree/octree: in these cases, we discussed a local L^2 projection mechanism. We also considered preserving the mesh topology—i.e., forcing cell merging decisions for one mesh at one time step to be the same as the mesh at the previous time step—with the goal of allowing temporal derivatives to be computed with high-order spatial accuracy, and also to eliminate unnecessary changes in mesh layout when, for example, the interface oscillates on either side of the volume fraction threshold.

In part two, several applications of the implicit mesh dG framework were presented concerning incompressible fluid flow, including single phase flow in non-trivial geometry, two phase flow driven by surface tension, rigid body fluid–structure interaction, and free surface flow. In these applications, the framework was coupled to a class of methods known as interfacial gauge methods [8] for solving the associated incompressible Navier–Stokes equations. These methods, which are similar in some aspects to archetypical projection methods, allow high-order accurate schemes to be developed more easily as the numerical coupling between fluid velocity, pressure, and interface location is weaker. A predictor–corrector time stepping algorithm was designed for each of these applications, and by means of numerous convergence tests examining convergence in the maximum norm, it was shown that the time stepping methods are second-order accurate, while high-order spatial accuracy is achieved in all physical quantities of interest, i.e., fluid velocity, pressure, and interface position. For single phase flow and rigid body fluid–structure interaction, the observed spatial order of accuracy is p; for two phase flow driven by surface tension, the spatial order depends on the parity of p (attributed to the numerical method for calculating curvature) such that, for $p=2$ and $p=3$, one obtains second order accuracy, and for $p=4$, fourth order accuracy is attained; last, for free surface flow, the order of accuracy is approximately 1.5 for $p=2$, order 2 for $p=3$, and order 3.5 for $p=4$.

Also presented were example problems demonstrating the utility of the framework. Two of these, i.e., high Reynolds number soap bubble oscillation, and Plateau–Rayleigh induced water ripples in free surface flow, also served as a form of experimental validation. With numerical results comparing favourably to experimental results, these examples also revealed physical phenomena not visible in the experiments and highlight some of the main motivational points of this work—to demonstrate the wide range of spatial scales often at play in fluid interface dynamics, and that high-order accurate methods can be used to examine how small-scale interfacial features develop and subsequently affect macroscopic dynamics.

Of note are some additional aspects relating to the framework's implementation. The three-dimensional problems and some of the more challenging two-dimensional problems benefit from high performance computing resources—to this end, the dG framework was parallelised with a standard domain decomposition approach and MPI. In particular, METIS [27] was used to perform the domain partitioning, and, since all of the solution algorithms are local, i.e., involve only stencil-based operations, standard synchronisation strategies involving cellular/elemental ghost layers can be used. As an additional note, regarding the implementation of interfacial gauge methods, during the course of a simulation it can sometimes be beneficial to occasionally perform a reset or reinitialisation, in which $\mathbf{m}$ is replaced by $\mathbf{u}$ and φ is set to zero. The reason for doing so is that, depending on the dynamics taking place, as $\mathbf{m}$ evolves it may develop finer spatial scales than those exhibited in the physical velocity field $\mathbf{u}$, which may subsequently require unnecessary spatial resolution. See [8] for further remarks

on this and the similarities and distinctions interfacial gauge methods have in comparison to traditional projection methods for incompressible fluid flow.

In the presentation, we have focused solely on problems in which topological changes in the interface do not occur. Implicit interface methods, e.g., the level set method, naturally and automatically handle topological changes such as droplet formation or bubble merging. However, simplistic cell merging algorithms may fail during topological changes, as small phase cells may be encountered that have no suitable neighbouring cells with which to merge, e.g., a small fluid droplet exhibiting a curvature length scale smaller than the length scale of the cells of the background quadtree/octree. Designing cell merging algorithms to construct implicitly defined meshes which robustly and accurately handle these cases may involve some intricacies. Possibilities include: locally refining the quadtree/octree; overriding the cell classification, i.e., small phase cells without suitable neighbours are reclassified as large; locally decreasing the element polynomial degree as in [28]; or searching wider afield than just the nearest-neighbour cells. This aspect will be the topic of future work on implicit mesh dG methods. Nevertheless, the cell merging procedure described in §2.1 of part one [1] never failed for any of the examples presented in this work.

A wide variety of possibilities exist for future work. A second-order predictor–corrector time stepping method was sufficient for the incompressible fluid flow problems considered here—for single phase flow, higher-order temporal schemes based on gauge methods and spectral deferred correction methods have been developed [12,14,17], and extension of such methods to interfacial fluid flow are likely possible. The presented geometric multigrid algorithms were based on h-multigrid (i.e., mesh hierarchies with progressively coarser implicitly defined meshes) and displayed optimal complexity, however their performance somewhat declined for high-order elements—a p-multigrid method for implicitly defined meshes, i.e., one in which the polynomial degree of the finite element space is progressively reduced, could be designed and is likely to yield improved efficiency for high order elements. Shown in §2.3.6 of part one [1] was a multiphase example, in which the implicitly defined mesh had interconnected interfaces and junctions—extension of the fluid flow solvers to multiple phases is also possible, in which case, instead of using the level set method to track the moving interface, one would use instead a multiphase algorithm such as the Voronoi implicit interface method [6,7]. Last, note that the developed implicit mesh dG framework could also be applied to a wide variety of other situations not considered here, including: generalised interfacial forces, such as Marangoni forces arising from non-uniform surfactant concentration; other types of fluid–structure interaction, such as deforming bodies and elastic interfaces; as well as non-Newtonian fluid flow.

Acknowledgements

This research was supported by a Luis W. Alvarez Postdoctoral Fellowship at Lawrence Berkeley National Laboratory, the Laboratory Directed Research and Development Program of LBNL, and by the Applied Mathematics Program of the U.S.DOE Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231. Some computations used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Appendix A. Axisymmetric modelling

In this section, we consider how to adapt the implicit mesh dG framework to the case of axisymmetric problems. To impose axisymmetry, it is natural to (i) use a cylindrical coordinate system, $r \in [0, \infty)$ and $z \in \mathbb{R}$, (ii) employ a quadtree background grid, and (iii) define the implicit mesh in the same way as one would for Cartesian coordinates. For simplicity of implementation, the nodal basis can be left unaltered, i.e., the standard tensor-product Gauss–Lobatto nodal basis can be used.[4] With this formulation, the primary modification needed within the dG methods (as compared to those in §2.3 of part one [1]) is to include an r Jacobian factor in all of the integration rules, e.g., in every mass matrix, and every face quadrature scheme. For curved elements, interphase faces, and intraphase faces cut by the interface, since the level set function $\phi : \Omega \to \mathbb{R}$ is defined in cylindrical coordinates, one can (i) use without modification the high-order quadrature schemes in [29] to compute a quadrature scheme with weights w_i and nodes (r_i, z_i), and then (ii) simply replace the computed weights with $w_i r_i$. For rectangular elements or faces, it is straightforward to adapt the integration rules to exactly account for the r Jacobian factor (for polynomial integrands). For example, one can show that the mass matrix for a rectangular element having extent $(r, r + 2\delta_r) \times (z, z + 2\delta_z)$ has the form

$$\left(\delta_r^2 M_r + (r + \delta_r)\delta_r M_0\right) \otimes \left(\delta_z M_0\right),$$

where M_0 is the one-dimensional reference mass matrix, $M_{0,ij} = \int_{-1}^{1} \ell_i(x)\ell_j(x)\,dx$, M_r is the one-dimensional reference mass matrix with a linear Jacobian factor, $M_{r,ij} = \int_{-1}^{1} \ell_i(x)\ell_j(x)x\,dx$, ℓ_i is the ith nodal basis function associated with the ith Lobatto node on the interval $[-1, 1]$, and $\otimes$ denotes the tensor product. Note that the matrices M_0 and M_r can be precomputed, allowing the mass matrix for all rectangular elements to be computed efficiently to machine-precision accuracy.

[4] Although the introduction of an r Jacobian factor alters the conditioning, it was found that the Lobatto basis continues to be well-conditioned for moderate p.

With the inclusion of the Jacobian factor in the integration rules, almost all of the discrete differential operators defined in §2.3 of part one [1] conveniently and automatically compute the correct operator in cylindrical coordinates. For example, the discrete (scalar) Laplacian correctly implements $\Delta = \frac{1}{r}\partial_r(r\partial_r) + \partial_{zz}$, and the discrete gradient correctly implements (∂_r, ∂_z). In particular, the discrete divergence, being the negative adjoint of the gradient, correctly computes $\frac{1}{r}\partial_r(ru) + \partial_z v$: one can think of the $\frac{1}{r}$ prefactor as coming from the inverse mass matrix in the formula for the discrete divergence, i.e., $M^{-1}G^{\mathsf{T}}M$, where M is the block-diagonal mass matrix. Furthermore, the advection operator $\nabla\cdot(\mathbf{u}\cdot)$ correctly computes the divergence in the case that $\cdot$ is a scalar (e.g., as in $\nabla\cdot(\mathbf{u}\phi)$), and also correctly computes the tensor divergence when $\cdot$ is a vector field (e.g., as in $\nabla\cdot(\mathbf{uu})$). However, note that the *vector* Laplacian operator requires special treatment, because, unlike in Cartesian coordinates, the vector Laplacian operator does not coincide with the scalar Laplacian applied to each component of the vector field. Indeed, letting $\mathbf{u} = (u, v)$, we have that $\Delta\mathbf{u} = (\Delta u - \frac{1}{r^2}u, \Delta v)$. Thus, to correctly solve a vector-valued elliptic interface problem, one must appropriately incorporate a correction of the form $\frac{1}{r^2}u$ to the corresponding discrete operator.

To discuss this further, consider the following elliptic interface problem for a vector field,

$$\begin{cases} -\Delta\mathbf{u} = \mathbf{f} & \text{in } \Omega_i \\ [\mathbf{u}] = \mathbf{g}_{ij} & \text{on } \Gamma_{ij} \\ [\partial_n\mathbf{u}] = \mathbf{h}_{ij} & \text{on } \Gamma_{ij} \\ \mathbf{u} = \mathbf{g} & \text{on } \Gamma_D \\ \partial_n\mathbf{u} = \mathbf{h} & \text{on } \Gamma_N. \end{cases} \tag{22}$$

In order to guarantee a physically reasonable solution, one must impose extra conditions on the solution at the origin, i.e., on the boundary $\partial\Omega \cap \{r = 0\}$. In particular, we assume that neither Γ_D nor Γ_N intersect $\{r = 0\}$, and require that (i) both u and v are even functions of r at $r = 0$, i.e. $\partial_r u = \partial_r v = 0$ at $r = 0$, and, in addition require that (ii) $u = 0$ at $r = 0$. Combined, these extra conditions eliminate the singular modes of the vector Laplacian; physically, the conditions correspond to the property that the radial component of the fluid velocity field must be zero at the origin, and that both the radial and vertical components must be even functions of r, as demanded by the smoothing effects of viscosity. In practice, the conditions in (i) are automatically enforced by the discrete solution, and do not require special attention; on the other hand, in order to ensure a well-behaved discrete operator (for the purposes of multigrid preconditioned conjugate gradient) the condition in (ii) needs to be imposed with a form of penalty parameter, as discussed shortly.

To solve (22), we solve two scalar problems,

$$\begin{cases} -\Delta u + \frac{1}{r^2}u = \mathbf{f}\cdot\hat{r} & \text{in } \Omega_i \\ [u] = \mathbf{g}_{ij}\cdot\hat{r} & \text{on } \Gamma_{ij} \\ [\partial_n u] = \mathbf{h}_{ij}\cdot\hat{r} & \text{on } \Gamma_{ij} \\ u = \mathbf{g}\cdot\hat{r} & \text{on } \Gamma_D \\ \partial_n u = \mathbf{h}\cdot\hat{r} & \text{on } \Gamma_N \\ \partial_r u = u = 0 & \text{on } \partial\Omega \cap \{r = 0\}, \end{cases} \quad \text{and} \quad \begin{cases} -\Delta v = \mathbf{f}\cdot\hat{z} & \text{in } \Omega_i \\ [v] = \mathbf{g}_{ij}\cdot\hat{z} & \text{on } \Gamma_{ij} \\ [\partial_n v] = \mathbf{h}_{ij}\cdot\hat{z} & \text{on } \Gamma_{ij} \\ v = \mathbf{g}\cdot\hat{z} & \text{on } \Gamma_D \\ \partial_n v = \mathbf{h}\cdot\hat{z} & \text{on } \Gamma_N \\ \partial_r v = 0 & \text{on } \partial\Omega \cap \{r = 0\}. \end{cases} \tag{23}$$

The second of these scalar problems can be solved with an essentially unmodified implementation of the algorithms in §2.3 of part one [1], once the r Jacobian factor is included in the integration rules: the treatment of the boundary conditions and interfacial jump conditions remains the same, the resulting linear system is symmetric positive (semi)definite, and the multigrid preconditioned conjugate gradient method outlined in §2.4 of part one [1] exhibits the same convergence efficiency. As for the first scalar problem, evidently we must modify the linear operator to include the $\frac{1}{r^2}u$ term. Variationally, this term corresponds to the bilinear form $(u, v) \mapsto \int_\Omega (\frac{1}{r^2}uv)r$, where the second r factor comes from the Jacobian. Note that the bilinear form is symmetric positive definite. Discretely, it would be of benefit to preserve this property, so that one can continue to use multigrid preconditioned conjugate gradient algorithms. Among a variety of different methods trialled as part of this work, it was found that the following scheme is effective, both in the efficiency of multigrid and in achieving high-order accuracy: the scheme approximates the bilinear form as

$$\int_\Omega (\tfrac{1}{r^2}uv)r \approx \int_\Omega \mathcal{I}(f(u,r))\mathcal{I}(f(v,r))r = v^{\mathsf{T}}\mathcal{A}M\mathcal{A}u,$$

where $\mathcal{I}(f(u,r))$ is the nodal interpolant of $f(u,r)$, $\mathcal{A}$ is the diagonal matrix whose diagonal entry for node i is $f(1, r_i)$, and $f(\cdot,\cdot)$ is a fraction function that suitably accounts for division by zero, defined as

$$f(u,r) = \begin{cases} \frac{u}{r} & \text{if } r > 0, \\ \lambda & \text{otherwise,} \end{cases}$$

where $\lambda > 0$ is a fixed, user-defined penalty parameter. The role of f is to replace u with the nodal interpolant of u/r for the purposes of calculating the variational form for the term $\frac{1}{r^2}u$ in (23). At the origin, the penalty parameter effectively adds a contribution to the bilinear form similar to $\lambda\int_{\{r=0\}\cap\partial\Omega} uv$: note that it functionally serves the same purpose as that

of the penalty parameters used by the LDG methods of §2.3 in part one [1] to weakly impose Dirichlet boundary conditions. With this approximation, the matrix for the discretisation of the first system in (23) is precisely that given by equation (15) of part one [1] plus the extra term, and is given by

$$\Big(\sum_{1\le k\le 2} G_k^{\mathsf{T}} M G_k\Big) + \tau_0 A_0 + \sum_{i>j} \tau_{ij} A_{ij} + \tau_D A_D + \mathcal{A}M\mathcal{A}.$$

Note that the additional term, $\mathcal{A}M\mathcal{A}$, is symmetric positive definite, and so the entire matrix is symmetric positive definite. In essence, the penalty parameter λ weakly enforces the boundary condition $u = 0$ at $r = 0$. It should be chosen large enough to ensure the solution is high-order accurate, and also to ensure multigrid relaxation imposes the boundary condition in as few sweeps as possible, yet not too large to affect the conditioning of the overall linear system; in this work the parameter is fixed to be $\lambda = 10^6$, determined empirically to satisfy all three objectives. Indeed, a variety of convergence tests, with and without interfaces, as well as using Bessel functions of the first kind for exact solutions (being eigenmodes of the Laplacian operator in cylindrical coordinates, to confirm correct implementation), show that optimal order accuracy is attained, in the maximum norm, for all p tested, $1 \le p \le 4$.

Finally, we consider the case of the stress tensor divergence operator, $\nabla \cdot (\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathsf{T}})$. Recall that, in §2.3.5 of part one [1], a Cartesian coordinate system is assumed in order to derive the associated discrete operator as the divergence of each row of the matrix $\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathsf{T}}$. In cylindrical coordinates, the same operation can be used, provided a correction term of the form $2\frac{u}{r^2}$ is included in the first component of the resulting vector equation. Using the same technique as above—assuming for simplicity of presentation a diffusion coefficient α equal to unity—this results in a linear system of the form

$$\begin{pmatrix} 2G_1^{\mathsf{T}} M G_1 + G_2^{\mathsf{T}} M G_2 + 2\mathcal{A}M\mathcal{A} & G_2^{\mathsf{T}} M G_1 \\ G_1^{\mathsf{T}} M G_2 & G_1^{\mathsf{T}} M G_1 + 2G_2^{\mathsf{T}} M G_2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cdots \\ \cdots \end{pmatrix},$$

where for brevity of presentation the LDG penalty terms have been excluded. One can show that the treatment of boundary conditions and interfacial jump conditions remains the same, and so, as in the case of the vector Laplacian, an essentially unmodified implementation of the algorithms for the stress tensor divergence elliptic interface problem can be used in cylindrical coordinates as well as for Cartesian coordinates, provided the $\mathcal{A}M\mathcal{A}$ correction is made, and the mass matrices and face integration rules are adapted to include the r Jacobian factor. Once again, the matrix is symmetric positive (semi)definite, and the performance of multigrid preconditioned conjugate gradient is commensurate with that in Cartesian coordinates. Convergence tests confirm that the method is also high-order accurate, exhibiting optimal-order accuracy in the maximum norm for all p tested.

Appendix B. Partition of unity methods

Recall that, within the implicit mesh dG framework, a multiphase function u is piecewise polynomial and usually exhibits discontinuities across intraphase faces (of magnitude equal to the spatial discretisation error), whilst on interphase faces, u may exhibit physically borne $\mathcal{O}(1)$ discontinuities. On occasion, it can be useful to convert such a function into a multiphase function, continuous throughout each phase, in a manner which preserves high-order accuracy. In order to remove the discontinuities across intraphase faces, it follows that one must perform some kind of smoothing. However, note that such a smoothing process cannot smooth across the interface, as knowledge of the jump of the function u and its derivatives across the interface is a priori unknown.

On a conforming mesh (i.e., one without hanging element vertices) and with only one phase, a particularly simple and oft-used technique for removing discontinuities is to define a (globally continuous) piecewise polynomial function via the usual nodal interpolant, where, for every node situated on the boundary of two or more elements, the average value of all such elements is used. In other words, multi-valued collocated nodes are collapsed to one value via averaging. However, for the implicit meshes considered in this work, this technique will not yield a continuous function because the nodal interpolant of an arbitrary continuous function is in general discontinuous (owing to the nonconforming element geometry with hanging vertices, see, e.g., Fig. 1). Thus, an alternative mechanism is required.

A partition of unity method is adopted here, based on the background quadtree/octree, as follows. Associated to each cell U_i of the tree is a non-negative weight function w_i with support contained in the local patch of cells having U_i at its centre. Specifically, let $\mathcal{N}_i$ denote the collection of cells neighbouring U_i (sharing a face, edge, or vertex), including U_i itself; in two dimensions, $\mathcal{N}_i$ is typically a 3×3 patch ($3 \times 3 \times 3$ patch in 3D), but may include more or less depending on the mesh refinement structure of the quadtree/octree. Let r_i denote the radius of the largest ball that can fit inside $\mathcal{N}_i$, such that the ball is centred at the centre x_i^c of U_i. Then, define the weight function $w_i : \mathbb{R}^d \to \mathbb{R}$ for cell U_i by

$$w_i(x) = \prod_{j=1}^{d} \omega\big((x_j - x_{i,j}^c)/r_i\big),$$

where $\omega : \mathbb{R} \to \mathbb{R}$ is the one-dimensional bump function

$$\omega(\theta) = \begin{cases} \exp(C\frac{1}{\theta^2-1}) & \text{if } |\theta| < 1, \\ 0 & \text{if } |\theta| \geq 1. \end{cases}$$

Here, the constant C controls the width of the bump function, and thus the amount of blending in the resulting partition of unity method; it is fixed to be $C = \frac{32}{9}$, found empirically to provide a good amount of smoothing (i.e., neither too sharp or broad).

These weight functions are used within a multiphase partition of unity interpolant, as follows. Each nonempty phase cell $U_i \cap \Omega_\chi$ (for some phase χ) has associated with it a function $f_{\chi,i} : \mathbb{R}^d \to \mathbb{R}$ (to be defined shortly). Then we define $f_\chi : \Omega_\chi \to \mathbb{R}$ such that

$$f_\chi(x) = \frac{\sum_i w_i(x) f_{\chi,i}(x)}{\sum_i w_i(x)}, \tag{24}$$

where it is understood that the summation over i considers only those i for which $U_i \cap \Omega_\chi$ is nonempty. The value of f_χ at each point x is thus a weighted average of the values of $f_{\chi,i}$ in a local patch of cells surrounding x, normalised by the total accumulated weight. Because the weight functions w_i are infinitely smooth, it follows that f_χ inherits the regularity of $f_{\chi,i}$; in fact, in the following, $f_{\chi,i}$ are polynomials and so f_χ is a C^∞ function.

What remains is to define the functions $f_{\chi,i}$ that are associated with each nonempty phase cell $U_i \cap \Omega_\chi$. These are ultimately determined by the given elemental piecewise polynomial function $u \in V_h$ defined on the implicit mesh. It is necessary for the functions $f_{\chi,i}$ to be well-behaved everywhere in the support of their associated weight function, which means $f_{\chi,i}$ will be evaluated outside of U_i. Thus, while it is tempting to define $f_{\chi,i}$ as the polynomial coinciding with u restricted to $U_i \cap \Omega_\chi$, this is inadvisable in practice because doing so will entail the extrapolation of high-degree polynomials. We instead define $f_{\chi,i}$ as the best polynomial, in the sense of the L^2 norm, matching u in the neighbourhood $\mathcal{N}_i$ of U_i, i.e.,

$$f_{\chi,i} = \operatorname*{arg\,min}_{p \in \mathcal{P}} \sum_{U_j \in \mathcal{N}_i} C_j \int_{U_j \cap \Omega_\chi} (u - p)^2, \tag{25}$$

where $\mathcal{P}$ is the space of tensor product polynomials employed by the dG method, and C_j are weighting prefactors.[5] Equation (25) represents a simple positive definite quadratic optimisation problem, the solution of which can be computed with mass matrices, the precomputed quadrature schemes for curved cells, and a Cholesky factorisation, very similar to the L^2 projections considered elsewhere in this work. The output is a polynomial $f_{\chi,i}$ which locally best matches the given function u restricted to phase χ.

In summary, this partition of unity algorithm takes as input a piecewise polynomial function u defined on the elements of an implicit mesh, and outputs a piecewise polynomial function $f_{\chi,i}$ defined on the cells of the quadtree/octree (one polynomial per phase in the case of cells containing the interface), such that, when combined with the weighting in (24), results in a C^∞ function on each phase. The method is applicable to unstructured meshes, as well as multiple phases and curved domain geometry. Convergence tests confirm that the resulting partition of unity method is high-order accurate (order $p+1$) in the maximum norm. The functions $f_{\chi,i}$ can be easily computed in parallel (e.g., by using typical domain decomposition methods), and the evaluation of $f_\chi(x)$ in (24) can be made efficient by precomputing the list of neighbouring cells of each cell U_i.

Appendix C. Evaluation of interface mean curvature

The evaluation of interface mean curvature $\kappa = \nabla_s \cdot \mathbf{n}$ plays a key role in surface tension-driven dynamics. In particular, it is not κ itself which drives the motion, rather it is the gradient of κ (in analogy with the fluid pressure p). Furthermore, as the curvature influences the flow dynamics, and thus the evolution of the level set function itself, there is a highly non-linear coupling between the errors in computing κ and the errors in the evolving level set function ϕ.

One particular aspect affecting this coupling is that κ is a non-linear second-order differential having the propensity to greatly amplify any perturbations within a level set function. This observation applies to finite difference methods, but is especially exaggerated for high-order discontinuous Galerkin methods, wherein highly oscillatory modes (albeit small) are introduced into the level set function as it is being advected. As a result, it is insufficient to simply evaluate the curvature of each element's ϕ polynomial; one must incorporate neighbouring element information in a way that effectively dampens the influence of these high-order modes. Among a wide variety of methods trialled as part of this work, including filtering, variation diminishing methods, L^2 optimisation, and applying a discrete divergence to a normalised discrete gradient (to

[5] $C_j = 1$ for all j results in a weighting for which each cell in the patch is considered equally. Experiments indicated that a larger weight on the central cell can sometimes increase the accuracy of the partition of unity method, though this has not been rigorously explored; in this work, $C_i = 1$ for the central cell and $C_j = 0.5$ for all $j \neq i$, for neighbouring cells.

approximate $\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}$), the following method was found to be high-order accurate, stable, and insensitive to the aforementioned "noise." The approach consists of three steps, the first and last of which use the partition of unity method described in the previous section: given an elemental level set function $\phi \in V_h$,

1. Evaluate, on each cell U_i of the quadtree/octree, the function $\phi_{\chi,i}$ defined in (25) for each relevant phase χ (e.g., cells containing the interface are multivalued). In other words, for each cell, find the best polynomial (in the sense of L^2) that approximates ϕ in the small patch neighbourhood of the cell. This procedure dampens the small amplitude high-order modes in ϕ (which tend to be uncorrelated on adjacent elements) by favouring the accuracy of low-order modes. Note that this process does not smooth across the interface itself, owing to the construction of the partition of unity method—this is important because, as the level set function is evolved by the physical velocity $\mathbf{u}$, it will develop physically-justifiable discontinuities in its derivatives across the interface.
2. Define the piecewise polynomial function $k \in V_h$ as the nodal interpolant of $\kappa(\phi_{\chi,i})$, i.e., for each element $E \in \mathcal{E}$, for each node x of E, set $k|_E(x) = \nabla \cdot \frac{\nabla \phi_{\chi(E),i}}{|\nabla \phi_{\chi(E),i}|}$, where i is the parent cell of element E. Note that evaluating the curvature of a polynomial (here $\phi_{\chi(E),i}$) is straightforward,[6] only requiring the gradient and Hessian matrix of the polynomial.
3. Use the partition of unity method to smooth k; this results in a function $\kappa_{\mathrm{PU}} : (x, \chi) \mapsto \mathbb{R}$ which computes $\kappa(\phi)$ at the point x in phase χ. It is this functional which is evaluated as part of interphase face quadrature schemes to determine the interfacial jump source terms in the elliptic interface PDEs. In particular, for an interface Γ_{ij} between phases i and j, the average of the two computed curvatures is lifted, i.e., the function $x \mapsto \frac{1}{2}\kappa_{\mathrm{PU}}(x, i) + \frac{1}{2}\kappa_{\mathrm{PU}}(x, j)$ is used within the numerical quadrature schemes to calculate the source terms in the equivalent of equation (14) in part one [1].

Thus, this procedure involves two steps of smoothing: first to smooth the level set function ϕ by damping its high-order oscillatory modes, and then to smooth the computed mean curvature of the smoothed ϕ. One may interpret this as the action of a stencil, where the stencil is approximately 5×5 (in 2D) or $5 \times 5 \times 5$ (in 3D) in extent. Since the operation involves computing a second-order differential, it follows that at most two orders of accuracy are lost. In fact, convergence tests indicate that the order of accuracy observed in practice depends on the parity of p: if p is even, then the level set function is order $p+1$ accurate and its computed curvature is order p accurate; if p is odd, the level set function remains order $p+1$ accurate, but its computed curvature is order $p-1$ accurate. Thus, while neither case returns optimal order accuracy, it is in some sense favourable that one order of accuracy is gained when p is even.

As remarked, this approach was found in practice to be reliable and insensitive to the small-amplitude high-order modes in the evolving level set function. Because implicitly defined meshes involve a somewhat unstructured mesh topology (as compared to say, a uniform Cartesian grid, or a highly structured triangular/tetrahedral mesh) it appears unlikely that one could exploit the mesh topology to increase the order of accuracy. (See also the recent work of Kummer and Warburton [30], which examines alternative methods based on filtering operations.) An interesting avenue of study would be to examine this aspect further, and although it appears challenging, to design an algorithm to calculate curvature with optimal-order accuracy.

References

[1] R. Saye, Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part I, J. Comput. Phys. 344 (2017) 647–682, http://dx.doi.org/10.1016/j.jcp.2017.04.076.
[2] D.N. Arnold, F. Brezzi, B. Cockburn, L.D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (5) (2002) 1749–1779, http://dx.doi.org/10.1137/S0036142901384162.
[3] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1) (1988) 12–49, http://dx.doi.org/10.1016/0021-9991(88)90002-2.
[4] J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences, Cambridge University Press, 1999.
[5] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, 2003.
[6] R.I. Saye, J.A. Sethian, The Voronoi implicit interface method for computing multiphase physics, Proc. Natl. Acad. Sci. 108 (49) (2011) 19498–19503, http://dx.doi.org/10.1073/pnas.1111557108.
[7] R.I. Saye, J.A. Sethian, Analysis and applications of the Voronoi implicit interface method, J. Comput. Phys. 231 (18) (2012) 6051–6085, http://dx.doi.org/10.1016/j.jcp.2012.04.004.
[8] R. Saye, Interfacial gauge methods for incompressible fluid dynamics, Sci. Adv. 2 (6) (2016), http://dx.doi.org/10.1126/sciadv.1501869.
[9] A.J. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comput. 22 (104) (1968) 745–762, http://dx.doi.org/10.1090/S0025-5718-1968-0242392-2.

[6] In cylindrical coordinates and assuming axisymmetry, the formula for mean curvature is

$$\kappa = \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} = \frac{\phi_r}{r|\nabla \phi|} + \frac{\phi_{rr} + \phi_{zz}}{|\nabla \phi|} - \frac{\phi_r^2 \phi_{rr} + 2\phi_r \phi_{rz} \phi_z + \phi_z^2 \phi_{zz}}{|\nabla \phi|^3}.$$

In theory, the $\phi_r/(r|\nabla \phi|)$ term has a removal singularity at the origin, since for physically reasonable solutions, ϕ_r must be zero at $r = 0$; in practice, discretisation errors result in a strong singularity, i.e., the possibility of division by zero. To circumvent this issue, near the origin one can transition to a Taylor series approximation to evaluate $\phi_r/(r|\nabla \phi|) \approx \phi_{rr}/|\nabla \phi|$. For a sufficiently small transition zone, the resulting approximation remains high-order accurate.

[10] V.I. Oseledets, On a new way of writing the Navier–Stokes equation. The Hamiltonian formalism, Russ. Math. Surv. 44 (3) (1989) 210–211, http://dx.doi.org/10.1070/RM1989v044n03ABEH002122 (translated from Comm. Moscow Math. Soc. 1988).
[11] P.H. Roberts, A Hamiltonian theory for weakly interacting vortices, Mathematika 19 (2) (1972) 169–179, http://dx.doi.org/10.1112/S0025579300005611.
[12] A.S. Almgren, A.J. Aspden, J.B. Bell, M.L. Minion, On the use of higher-order projection methods for incompressible turbulent flow, SIAM J. Sci. Comput. 35 (1) (2013) B25–B42, http://dx.doi.org/10.1137/110829386.
[13] S.Y. Kadioglu, R. Klein, M.L. Minion, A fourth-order auxiliary variable projection method for zero-Mach number gas dynamics, J. Comput. Phys. 227 (3) (2008) 2012–2043, http://dx.doi.org/10.1016/j.jcp.2007.10.008.
[14] M.L. Minion, Semi-implicit projection methods for incompressible flow based on spectral deferred corrections, Appl. Numer. Math. 48 (2004) 369–387, http://dx.doi.org/10.1016/j.apnum.2003.11.005.
[15] J.L. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Comput. Methods Appl. Mech. Eng. 195 (2006) 6011–6045, http://dx.doi.org/10.1016/j.cma.2005.10.010.
[16] J.-G. Liu, J. Liu, R.L. Pego, Stable and accurate pressure approximation for unsteady incompressible viscous flow, J. Comput. Phys. 229 (9) (2010) 3428–3453, http://dx.doi.org/10.1016/j.jcp.2010.01.010.
[17] R.I. Saye, M.L. Minion, 2017, in preparation.
[18] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (2) (1985) 308–323, http://dx.doi.org/10.1016/0021-9991(85)90148-2.
[19] D.L. Brown, R. Cortez, M.L. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2) (2001) 464–499, http://dx.doi.org/10.1006/jcph.2001.6715.
[20] R.I. Saye, High-order methods for computing distances to implicitly defined surfaces, Commun. Appl. Math. Comput. Sci. 9 (1) (2014) 107–141, http://dx.doi.org/10.2140/camcos.2014.9.107.
[21] M.M. Francois, S.J. Cummins, E.D. Dendy, D.B. Kothe, J.M. Sicilian, M.W. Williams, A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework, J. Comput. Phys. 213 (1) (2006) 141–173, http://dx.doi.org/10.1016/j.jcp.2005.08.004.
[22] J.U. Brackbill, D.B. Kothe, C. Zemach, A continuum method for modeling surface tension, J. Comput. Phys. 100 (2) (1992) 335–354, http://dx.doi.org/10.1016/0021-9991(92)90240-Y.
[23] J.S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Springer, 2008.
[24] B. Preskill, The Jump Splice for Elliptic Interface Problems and the Incompressible Navier–Stokes Equations, Ph.D. thesis, University of California, Berkeley, 2015.
[25] U. Kornek, F. Müller, K. Harth, A. Hahn, S. Ganesan, L. Tobiska, R. Stannarius, Oscillations of soap bubbles, New J. Phys. 12 (7) (2010) 073031, http://dx.doi.org/10.1088/1367-2630/12/7/073031.
[26] M.J. Hancock, J.W.M. Bush, Fluid pipes, J. Fluid Mech. 466 (2002) 285–304, http://dx.doi.org/10.1017/S0022112002001258.
[27] G. Karypis, V. Kumar, A fast and highly quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1) (1999) 359–392, http://dx.doi.org/10.1137/S1064827595287997.
[28] A. Fröhlcke, A Boundary Conformal Discontinuous Galerkin Method for Electromagnetic Field Problems on Cartesian Grids, Ph.D. thesis, Technische Universität, Darmstadt, 2013.
[29] R.I. Saye, High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles, SIAM J. Sci. Comput. 37 (2) (2015) A993–A1019, http://dx.doi.org/10.1137/140966290.
[30] F. Kummer, T. Warburton, Patch-recovery filters for curvature in discontinuous Galerkin-based level-set methods, Commun. Comput. Phys. 19 (2) (2016) 329–353, http://dx.doi.org/10.4208/cicp.191114.140715a.