Journal of Computational Physics ••• (••••) •••–•••

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 I

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 xxxx

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.2017.04.076

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

A panoply of fluid dynamics problems involve surface, boundary, and interface motion playing a pivotal role in the global dynamics. Examples include transport of solvents in bubble aeration, rupture of thin films in foam dynamics, droplet atomisation in spray painting devices, and the design of propeller blades. When modelling these problems computationally, a careful and precise treatment of the boundary and interface motion is often necessary—small boundary layers near fluid–air or fluid–fluid interfaces can strikingly affect their evolving shape, while small-scale features in interface or boundary geometry can affect fluid dynamics far afield.

In this work, 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. The implicit representation of moving surfaces offers a variety of benefits: 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. Combined with the flexibility dG methods have in working with unstructured meshes, the framework is conceptually straightforward: essentially all that is required to compute quadrature rules for the curved elements and faces which are implicitly defined by the interface or boundary.

Originally inspired by the work of Johansson and Larson [1], the implicit mesh dG framework developed here was briefly introduced as part of prior work on numerical methods for interfacial fluid flow [2]. In this two-part paper, the framework is presented in detail, together with several applications to incompressible fluid flow. In particular, in part one we discuss:

- Multiphase implicitly defined meshes;
- Quadrature for implicitly defined elements and faces, to accurately compute elemental mass matrices, face integration rules, L^2 projections, etc.;
- DG methods to solve scalar and vector-valued elliptic interface partial differential equations (PDEs) with interfacial jumps in ellipticity coefficient, resulting in symmetric positive (semi)definite linear systems;
- Design of projection operators used within incompressible fluid flow, which, except for the influence of small parameters, are discretely idempotent;
- Geometric multigrid methods to efficiently solve elliptic interface PDEs, including choices for mesh hierarchies and relaxation methods, and their use as preconditioners for the conjugate gradient method;
- Partition of unity methods to smoothly interpolate whilst preserving high-order accuracy; and
- Extension of the dG framework to cylindrical coordinate systems and axisymmetric problems;

Furthermore, in the context of evolving interface problems, we also discuss:

- Combining level set methods with implicitly defined meshes to obtain meshes that evolve in time;
- Transferring state (e.g., velocity field, pressure) between meshes as they evolve, handling element creation and destruction, whilst preserving high-order accuracy;
- Preserving mesh topology, by using fuzzy thresholds in the creation of implicit meshes, to accurately compute temporal derivatives; and
- Integration of time-dependent adaptive mesh refinement.

To demonstrate the utility of implicit mesh dG methods, several example applications are presented in part two—see [3]—concerning single phase flow in nontrivial geometry, two phase flow driven by surface tension, rigid body fluid–structure interaction, and free surface flow. These examples consider incompressible fluid flow and make use of recently developed *interfacial gauge methods* [2] to solve the corresponding Navier–Stokes equations. Convergence analyses are conducted throughout the work, verifying the high-order accuracy of the implicit mesh dG framework—convergence is generally

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

examined in the maximum/infinity norm, thereby ruling out "parasitic currents" and other types of numerical boundary layers which often diminish the accuracy of low-order methods. In particular:

- Two- and three-dimensional examples and convergence tests are presented.
- Optimal-complexity multigrid preconditioned conjugate gradient algorithms are demonstrated.
- Stable time stepping methods are designed for each application, wherein the usual CFL constraint applies, i.e., $\Delta t \leq C \min h/\|\mathbf{u}\|$, where $\|\mathbf{u}\|$ is the local fluid speed and C is approximately 0.05 to 0.2 in magnitude.
- Optimal-order accuracy is demonstrated when solving elliptic interface PDEs on curved geometries with multiple, curved, interconnected interfaces: for a degree p piecewise polynomial finite element space, the order of accuracy is $p+1$ in the maximum norm.
- For most of the incompressible fluid flow applications, the order of accuracy is one less than optimal, i.e., the error in the computed velocity field, pressure field, and interface location are all $\mathcal{O}(h^p)$ in the maximum norm, where h is the typical element size. (The reduced order of accuracy is attributed to associated projection operators on unstructured meshes.)

In the next section, we discuss some of the background and motivation underlying implicit mesh dG methods, followed by an outline for the remainder of this work.

1.1. Background and motivation

Within the various paradigms for solving PDEs on domains with changing geometry—including fixed-grid methods, moving mesh methods, and their hybrids such as arbitrary Lagrangian–Eulerian (ALE)—a wide variety of numerical methods have been developed to compute fluid interface dynamics. Prominent examples include: (i) Peskin's immersed boundary method [4,5], a mixed Eulerian–Lagrangian method in which interfacial forces are represented as fluid body forces via smoothed, discrete approximations of Dirac delta functions; (ii) LeVeque and Li's immersed interface method [6,7], a sharp interface finite difference approach in which finite difference stencils are adapted to include jump conditions; (iii) level set methods [8], in which interfacial forces are regularised with smoothed delta functions [9–11]; (iv) the ghost fluid method [12], a finite difference method which introduces extra grid-collocated degrees of freedom determined by extrapolation and jump conditions; and (v) the volume of fluid method [13], an Eulerian method in which the volume fraction of each fluid phase is tracked in each grid cell. These methods are typically implemented together with a fixed computational grid on which the PDEs governing the physics are also solved; other possibilities include: (i) moving mesh methods, in which the mesh stretches and contorts as its nodes follow the interface, see, e.g., [14]; (ii) finite element methods that enrich the solution space with basis functions accounting for interfacial discontinuities by locally adapting to the interface shape, such as the extended finite element method (XFEM) [15,16] and discontinuous Galerkin hybrids [17]; and (iii) cut-cell and embedded boundary finite volume and finite element methods, wherein the moving interface cuts through a background reference grid, subsequently defining a new mesh with changing cell shapes. The approach adopted in this work follows in part the idea of this latter category.

To illustrate, consider a uniform Cartesian grid together with an interface separating two fluids. Using the interface to cut through the cells of the grid, a mesh can be defined consisting mostly of rectangular elements away from the interface, together with elements corresponding to grid cells cut into pieces by the interface. The resulting mesh is automatically body conforming, i.e., the element faces align precisely with the interface geometry. Moreover, by using a simplistic background grid (Cartesian in this example), such meshes are often easier to define compared to relying on more traditional meshing mechanisms. Numerical techniques using this idea are variously known as *cut cell*, *unfitted*, *embedded boundary*, or *immersed boundary* methods, and are most often used in combination with finite volume and finite element methods. In practical settings, of critical consideration is the following property: cells cut by the interface an be arbitrarily small in size. Used without any kind of regularisation in a cut cell finite volume method or finite element method, tiny cut cells lead to significant numerical issues, such as very ill-conditioned discretisations of PDE problems (e.g., the condition number of the discrete Laplacian operator can be arbitrarily large), while time step constraints in evolutionary problems can be arbitrarily severe. Therefore, tiny cut cells must be appropriately accounted for and a variety of techniques have been developed to this end. One option is to locally scale discrete stencils with a weighting that depends on the area/volume of the corresponding cells, in such a way that the conditioning and stability is improved, see, e.g., [18–21]. An alternative strategy is to include penalty parameters in variational statements to weakly impose interfacial jump conditions and boundary conditions on the cut cell faces, such as in the Nitsche finite element method [22]. These methods provide a consistent mathematical framework for handling small cut cells, and can also be used to analyse coercivity in elliptic problems as well as stability and accuracy, see, e.g., work on Nitsche cut cell methods for the wave equation [23], solving Laplace–Beltrami equations on codimension-one surfaces cutting through a background grid [24], Stokes problems with phase-dependent viscosity and surface tension [25], and solving electrohydrodynamic potential flow in moving domains [26]; see also the review by Burman et al. on cut cell finite element and Nitsche methods [27].

The implicitly defined meshes used in this work are similar to cut cell methods, but the troublesome ill-conditioning issues arising from tiny cut cells are avoided by using a simple cell merging procedure. The strategy adopted here was originally inspired by the work of Johansson and Larson [1] where it complements the generality and flexibility of discon-

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

tinuous Galerkin methods. (For an introduction to dG methods, see, e.g., Hesthaven and Warburton [28].) In particular, cut cells which are identified as "tiny" (according to a user-defined threshold) are merged with neighbouring cells to define composite or "extended" elements (see, e.g., Fig. 1) ultimately forming the mesh of a dG method. In the cited work, a priori estimates on accuracy and conditioning are proven for a particular dG discretisation of Poisson's problem in a domain with implicitly defined boundary, and numerical results are shown demonstrating optimal-order accuracy for second and third order methods. By using high-order accurate quadrature schemes to compute the necessary element mass matrices and face integration rules for the implicitly defined elements, the same formulation can be made arbitrarily high-order accurate [29].

The idea of cell merging, also known as cell agglomeration and element extension, has also been used in a variety of other settings. In the dissertation of Hunt [30], cell merging is used in a three-dimensional finite volume method for inviscid compressible flow with moving boundaries. Later, in the dissertation of Fröhlcke [31], a boundary conformal dG method is used to solve three-dimensional quasi-electrostatic problems; developed at the same time as the work of Johansson and Larson, and also based on high-order dG methods, Fröhlcke considers various strategies for handling cut cells which, in addition to cell merging with neighbouring cells sharing the largest face, also considers a strategy of adaptive approximation order, i.e., reducing the polynomial degree in elements which are considered tiny. Müller et al. [32] employ agglomeration for a static, single phase domain to solve the two-dimensional compressible Navier–Stokes equations. Elliptic interface problems with discontinuous coefficients across embedded interfaces with corners, giving rise to possibly singular solutions, are considered in a cell merging dG method of Brandstetter and Govindjee [33]. As part of ongoing work (ultimately with the goal of addressing time-dependent interface evolution problems), Kummer [34] considers high-order accurate spatial discretisation for static, time-independent, two-phase incompressible Navier–Stokes equations, using cell agglomeration together with a high-order accurate quadrature method based hierarchical moment fitting [35]. Cell merging is also used by Antonietti et al. [36] as part of meshing domains with complex shapes: domain boundaries exhibiting a wide range of geometric scales are meshed with a hierarchy built on refinement in combination with an agglomeration procedure.

Cell merging provides a conceptually straightforward way to define a domain boundary- and interface-conforming mesh. In the present work, the geometry of the domain boundary and multiphase interfaces are defined implicitly. From a mathematical and analytical point of view, the resulting implicitly defined meshes can be used in a dG method essentially as-is, without knowledge of their specific construction. As such, instead of naming the developed dG framework as cut cell, or embedded boundary, or as an extended discontinuous Galerkin method, it is here simply referred to as dG methods which use implicitly defined meshes.

The discussion so far has largely focused on spatial considerations when designing accurate numerical methods for fluid interface dynamics, e.g., defining sharp interface methods which allow jump conditions (e.g., jumps in fluid pressure) to be captured with high-order accuracy. In the case of incompressible fluid flow, the incompressibility constraint is an additional aspect requiring special attention when designing high-order accurate methods. To solve the associated incompressible Navier–Stokes equations, a widely applied technique is that of projection methods, pioneered by Chorin [37]. These methods can be thought of as a fractional stepping method in which, at each time step, the Navier–Stokes momentum equations are used to compute an intermediate velocity field not necessarily satisfying the incompressibility constraint. This intermediate velocity field is then projected onto the space of divergence-free vector fields, thereby determining the fluid velocity $\mathbf{u}$ at the end of the time step. As elucidated by Brown, Cortez and Minion [38], this fractional stepping approach introduces a nonphysical coupling in the discrete velocity and pressure field, resulting in numerical boundary layers that limit the simplest of approaches to first order accuracy in the maximum norm. Owing to the robustness, wide utility, and computational efficiency of projection methods, the design of high-order accurate projection methods has consequently received extensive attention in the literature; see, e.g., the second order methods of Kim and Moin [39] and Brown et al. [38], third and higher order (single phase) projection methods based on consistent splitting schemes and slip correction schemes in the reviews [40,41], and high-order accurate schemes based on spectral deferred corrections [42–44]. Projection methods have also been applied to multiphase fluid flow in a wide variety of formulations, including the continuum surface force model of Brackbill et al. [9] and immersed boundary methods [4,5] which recast interfacial surface forces as body forces [10,11]; modified projection methods which build pressure jump conditions directly into the projection operator as in the immersed interface method [6,7] and ghost fluid method [12]; and projection methods which design various kinds of pressure predictors accounting for surface tension, see, e.g., [45,46]. However, complications can arise when applying standard projection method ideas to multiphase flow if the end goal is to capture or impose physically-relevant interfacial jump conditions with high-order accuracy. In a similar way that numerical boundary layers are created at the domain boundary (associated with the non-commutativity of the viscous diffusion and projection operators [38,40,41]), numerical boundary layers can also be created at the interface (see [2]) and these artefacts can be especially exaggerated in the case when different fluid phases have different densities or viscosities.[1] These numerical boundary layers, at the domain boundary in the case of single phase

[1] To briefly motivate the issues involved, consider a standard projection method for two-phase flow driven by surface tension. Given the velocity field $\mathbf{u}^n$ at time step n, the first step is to perform a viscous solve to find $\mathbf{u}^\star$ such that $\rho_i(\frac{1}{\Delta t}(\mathbf{u}^\star - \mathbf{u}^n) + (\mathbf{u}\cdot\nabla\mathbf{u})^{n+1/2}) = -\nabla q + \nabla\cdot(\mu_i(\nabla\mathbf{u}^n + \nabla\mathbf{u}^\star))$ in Ω_i (phase i), where q is some form of pressure predictor (possibly zero), and ρ_i and μ_i is the density and viscosity of phase i, subject to some set of interfacial jump conditions for $[\mathbf{u}^\star]$ and $[\mu(\nabla\mathbf{u}^\star + \nabla\mathbf{u}^{\star\mathsf{T}})\mathbf{n}]$ on the interface Γ; usually, it is required or implicitly assumed that $[\mathbf{u}^\star] = 0$. The second step is to project $\mathbf{u}^\star$ to find $\mathbf{u}^{n+1} = \mathbf{u}^\star - \rho_i^{-1}\nabla\phi$ in Ω_i, where $\nabla\cdot\rho_i^{-1}\nabla\phi = \nabla\cdot\mathbf{u}^\star$ in Ω_i, subject to suitable interfacial jump conditions for the elliptic interface problem for ϕ, i.e., some conditions on $[\phi]$ and $[\rho^{-1}\partial_n\phi]$ on Γ; usually, $[\rho^{-1}\partial_n\phi] = 0$. However, when $\rho_1 \neq \rho_2$, it cannot be guaranteed that the Poisson problem for ϕ admits a solution for which $[\rho^{-1}\nabla\phi] = 0$; this implies that $[\mathbf{u}^{n+1}]$ is not necessarily zero on the interface—this forms a direct analogy to the issue

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

flow, and adjacent to the interface in the case of multiphase flow, persist regardless of how high-order accurate the spatial discretisation may be.

An alternative formulation of the incompressible Navier–Stokes equations, first introduced by Oseledets [47] (see also [48]) for single phase flow, and extended to interfacial fluid flow in prior work [2], leads to a different class of methods, denoted *interfacial gauge methods*. Instead of solving directly for $\mathbf{u}$, these methods recast the equations to solve instead for an auxiliary vector field, denoted $\mathbf{m}$, whose divergence-free component gives the physical velocity $\mathbf{u}$. Compared to archetypical projection methods, interfacial gauge methods have a weaker coupling between fluid velocity, pressure, and interface dynamics, and allow stable and high-order accurate schemes to be developed more easily. These methods are briefly introduced later in this work; for more information regarding their design, the reader is referred to [2].

1.2. Outline

This paper is divided into two main parts—part one: development of the implicit mesh dG framework, and part two [3]: application of the framework to a variety of incompressible fluid flow problems. In part one:

(1) Implicitly defined meshes are introduced.
(2) Preliminaries of the dG framework are described, including notation, choice of nodal basis, jump operators, quadrature, mass matrix computation, and a more detailed algorithm for defining implicit meshes.
(3) Local dG methods for elliptic interface problems are designed, beginning with uniform-ellipticity scalar equations and then extending to variable-coefficient and vector-valued elliptic interface problems. Convergence results are presented, as is the design of projection operators for incompressible fluid flow.
(4) Geometric multigrid methods for elliptic interface problems are designed for use as preconditioners in conjugate gradient algorithms.
(5) Discretisation of the advection operator is discussed.
(6) Topics related to evolving interface problems are considered, including coupling interface dynamics to physics, high-order accurate reinitialisation, transferring state between evolving implicit meshes, and briefly on the integration of adaptive mesh refinement.

In the second part—see [3]—the implicit mesh dG framework is applied to:

(1) Single phase incompressible fluid flow, which also serves as an introduction to gauge methods.
(2) Two phase fluid flow driven by surface tension.
(3) Rigid body fluid–structure interaction.
(4) Free surface flow.

Part two concludes with a summary of the main ideas underlying the implicit mesh dG framework and possibilities for future work. In addition, the three appendices of part two 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. An implicit mesh discontinuous Galerkin framework

In the first part of this paper, we design a discontinuous Galerkin (dG) framework based on implicitly defined meshes, having in mind the goal of applying the framework to a variety of problems in fluid interface dynamics. First, the concept of implicitly defined meshes is introduced, followed by a discussion of the essential components of the dG method, such as quadrature, mass matrices, and definition of interphase faces, intraphase faces and jump operators. Next, we develop a set of high-order accurate discretisations for a variety of elliptic interface partial differential equations, demonstrating optimal-order accuracy with a series of convergence tests. Geometric multigrid methods, which can be used to efficiently solve the associated linear systems, are then discussed. Then, in the final section, we consider a variety of aspects concerning moving interface problems, including the coupling of interface dynamics to physics, defining evolving implicit meshes, transferring state between meshes, and finally, a brief discussion on possibilities for adaptive mesh refinement.

2.1. Implicitly defined meshes

An implicitly defined mesh involves the interaction of two geometric objects: (i) a background reference grid, whose geometric description is relatively simple, and (ii) an implicit description of any embedded interfaces and/or the domain boundary. In this work, regular Cartesian grids, quadtrees, and octrees are used for the reference grid, while the interfaces are defined by either the level set method (for two phases) or the Voronoi implicit interface method [49] (for multiple phases).

of enforcing the correct tangential boundary conditions for single phase projection methods, see [38,40,41]. Moreover, there may also be inconsistencies concerning the interfacial jump condition for the physical stress associated with the projected velocity field, where additional complications can arise in the case that $\mu_1 \neq \mu_2$. See [2] for further discussion.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

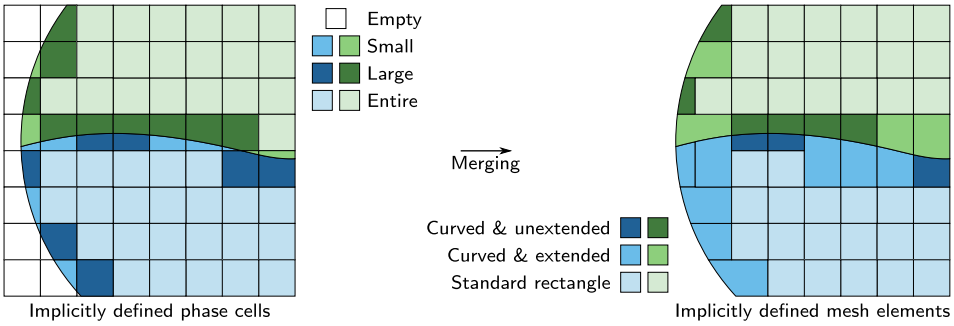

Fig. 1. Defining the mesh in the implicit mesh dG framework. (*left*) Phase cells, defined by the intersection of each phase (blue and green) with the cells of a background Cartesian/quadtree grid, are classified according to whether they fall entirely within one phase ("entire," light blue/green and 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.)

To simplify the presentation, the two-phase case is considered here. Let $\Omega \subset \mathbb{R}^d$ denote the domain, and let $\phi : \Omega \to \mathbb{R}$ be a level set function which implicitly defines the interface $\Gamma = \{x \in \Omega : \phi(x) = 0\}$. Denote by $\Omega_1 = \{x \in \Omega : \phi(x) < 0\}$ the region occupied by phase one and $\Omega_2 = \{x \in \Omega : \phi(x) > 0\}$ the region occupied by phase two. Let U be a rectangular domain which encloses all of Ω, and define a quadtree (in 2D) or octree (in 3D) grid such that $U = \bigcup_i U_i$, where U_i are rectangular cells. This defines a collection of *phase cells*, $U_i \cap \Omega_j$, most of which are rectangular and fall entirely within a single phase (denoted *entire* cells), whereas others intersect $\partial\Omega$ or Γ (*partial* cells), with the remainder falling outside of the domain (*empty* cells), see Fig. 1.

Cells which are deemed "small" are merged with nearby cells to ultimately form the elements of the implicitly defined mesh. A variety of possibilities exist for deciding whether a cell is small; in this work a simple strategy is adopted, whereby a phase cell $U_i \cap \Omega_j$ is viewed as small if its volume fraction is less than 40%, i.e., $|U_i \cap \Omega_j| < 0.4|U_i|$. Small cells are merged with neighbouring cells in the same phase according to the following scheme: of the cells sharing a face, edge, or vertex with the small cell, the cell which has the largest volume fraction is used. The result of this process is a collection of elements consisting of standard rectangular elements and elements with curved, implicitly defined boundaries; Fig. 1 shows a two-dimensional example. Some remarks are in order:

- Note that the geometry of the resulting mesh is never explicitly constructed or parameterised. Instead, the mesh is implicitly defined by the background grid and the implicitly defined interface. Thus, the curvature and shape of the interface is completely characterised by the level set function. In the context of the following discontinuous Galerkin framework, the geometry of the interface is inferred solely through the element and face quadrature rules.
- The mesh is automatically "body conforming": the interface is sharply represented by the faces between elements belonging to different phases. Subsequently, interfacial jump conditions can be imposed with high-order accuracy.
- Computation of the volume fraction of each cell $U_i \cap \Omega_j$ requires an algorithm capable of computing implicitly defined volume integrals. Although it is unnecessary to compute these volume fractions with a high degree of accuracy, it is nevertheless important to identify cells with non-zero volume. Traditional techniques, such as extracting the isosurface of ϕ in Ω as a polyhedron (for instance, as in marching cubes [50] or marching tetrahedra [51–53]), may fail in this regard. This aspect is discussed further in §2.2.2.
- The small volume fraction threshold, i.e., 40% in the above, is a user-chosen parameter. The smaller the fraction, the smaller the size of the extended elements, which can increase solution resolution and accuracy. However, if the threshold becomes too small, then a wider disparity in element size (i.e., the length scale "h" of each element) can occur, and for very small fractions, the local condition number can considerably worsen, causing discrete differential operators to become ill-conditioned and time step constraints more severe. This has a direct analogy with the problem of "tiny" cells in traditional cut-cell methodologies.

  In the present work, we do not examine the effect of a vanishingly small fraction threshold on conditioning and time step constraints. Instead, it is noted that thresholds between 20% and 50% perform well in a variety of circumstances (i.e., based on a variety of tests, the conditioning of elliptic interface problems is sufficiently mild for double-precision arithmetic, and perceived time step constraints for advective problems are commensurate of typical constraints occurring within discontinuous Galerkin methods), whereas fractions smaller than 10% become problematic, especially when using high-order elements. In particular, throughout the present work, the parameter of 40% is used, as it offers a good compromise between excessively large extended elements, while maintaining good conditioning for moderately high-order elements ($p \leq 5$) in both two and three dimensions. For further observations regarding the effect of the threshold on conditioning, see, e.g., Johansson and Larson [1] where its effect on elliptic PDE problems is analysed both theoretically and experimentally.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

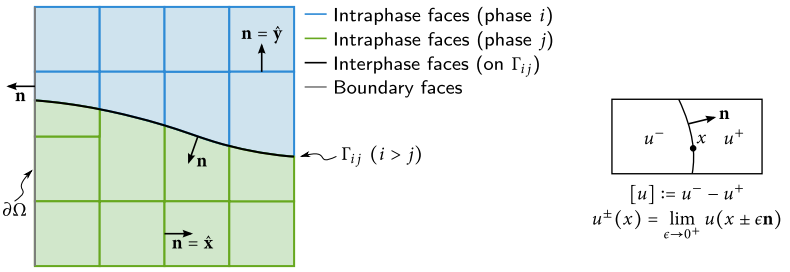

Fig. 2. Face nomenclature and definition of the jump operator $[\cdot]$. (*left*) A schematic of a two-phase mesh: intraphase faces are defined as faces shared by elements in the same phase, whereas interphase faces and boundary faces are located on Γ_{ij} and $\partial\Omega$, respectively. The orientation of the normal of the interface is such that $\mathbf{n}$ points from the phase with highest index i into the phase with lowest index j. (*right*) Two elements are separated by a face with the indicated normal $\mathbf{n}$; the jump of a function u is such that $[u] = u^- - u^+$, where "$-$" and "$+$" denotes the element on the "left" and "right", respectively, while $u^{\pm}$ denotes the values of the function upon restriction to the left and right element.

- Last, note that in some circumstances, the merging process may fail. In other words, a small phase cell may be identified having no suitable neighbours to merge with. This typically happens when the interface is under-resolved, such as an isolated droplet a small fraction of the cell size, or, during topological changes, such as merging or breaking of the interface. A variety of strategies could be adopted in such circumstances, e.g., temporarily weakening the fraction threshold or subdividing the cell locally as a form of adaptive mesh refinement. Conversely, provided the interface is sufficiently resolved, the merging process always succeeds: in particular, the merging process never failed for any of the results presented in this work.

In summary, the construction of an implicitly defined mesh takes as input: (i) a quadtree/octree background grid together with (ii) an implicit description of the interface; and outputs a collection of elements, $\mathcal{E} = \bigcup_i E_i$, each of which belongs in its entirety to a single phase j. Implicit in this description is a collection of faces representing the boundaries between elements, determined by (i) the cell boundaries of the quadtree/octree, (ii) the implicitly defined interface, and (iii) the domain boundary. These faces are either flat (when they arise from the cell boundaries of the background grid) or curved (when they arise from the implicitly defined interface or perhaps an implicitly defined domain boundary).

2.2. Preliminaries

In order to design an implicit mesh dG framework making use of implicitly defined meshes, we first define some notation and consider a variety of aspects which underpin much of the framework's construction.

2.2.1. Notation

We consider a multiphase domain $\Omega = \bigcup_i \Omega_i$ in $\mathbb{R}^d$, and let $\Gamma = \bigcup_{i>j} \Gamma_{ij}$ denote the interface, where $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$ is the interface between phase i and phase j. Let $\mathbf{n}$ denote the unit normal of the interface Γ_{ij}, pointing from Ω_i into Ω_j when $i > j$, or, in the case of the domain boundary, let $\mathbf{n}$ denote the unit outwards-pointing normal. Let $\mathcal{E}$ denote the set of elements of the mesh, and let $\chi(E)$ denote the phase of an element $E \in \mathcal{E}$, i.e. if $E \subseteq \Omega_i$, then $\chi(E) = i$.

In the discontinuous Galerkin framework, quantities of state (such as fluid velocity) are piecewise-polynomial. In this work, the background grid is a quadtree (in 2D) or octree (in 3D), and so most of the elements of the implicit mesh are rectangular. It is therefore natural to choose a tensor product polynomial space (though this is not the only possibility). Let $p \geq 1$ be an integer and define $\mathcal{Q}_p(E)$ to be the space of tensor product polynomials of degree p on the element E. For example, $\mathcal{Q}_3$ is the space of bicubic (in 2D) or tricubic (in 3D) polynomials having 16 or 64 degrees of freedom, respectively. Let V_h be the corresponding space of discontinuous piecewise polynomials on the mesh:

$$V_h := \left\{ u : \Omega \to \mathbb{R} \;\middle|\; u|_E \in \mathcal{Q}_p(E) \text{ for every } E \in \mathcal{E} \right\}.$$

Define also V_h^d to be the space of piecewise polynomial vector fields,

$$V_h^d := \left\{ \mathbf{u} : \Omega \to \mathbb{R}^d \;\middle|\; (\mathbf{u} \cdot \mathbf{e}_i)|_E \in \mathcal{Q}_p(E) \text{ for every } E \in \mathcal{E} \text{ and } 1 \leq i \leq d \right\}.$$

Here, $\mathbf{e}_i$ denotes the standard basis vector in the direction of the ith coordinate. The natural L^2 inner product on V_h is denoted by $(\cdot, \cdot)$, $\|\cdot\|$ denotes the corresponding norm, $\|u\|^2 = (u, u)$, with analogous definitions for V_h^d.

Regarding the faces of the mesh, define *intraphase faces* as those shared by two elements of the same phase; *interphase faces* as those shared by two elements of differing phases (and thus are situated on Γ_{ij} for some $i > j$); and *boundary faces* as those situated on $\partial\Omega$; see Fig. 2. Define also Γ_0 as the set of all points belonging to intraphase faces. Each face has a corresponding normal vector: intraphase faces, which are flat, lie in a particular coordinate plane and $\mathbf{n}$ is defined to point from "left-to-right", i.e., for vertical faces in 2D, $\mathbf{n} = \hat{\mathbf{x}}$, and for horizontal faces, $\mathbf{n} = \hat{\mathbf{y}}$. Interphase faces adopt the same normal vector as the interface Γ_{ij} on which they are situated, i.e. the normal of a interphase face points from

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

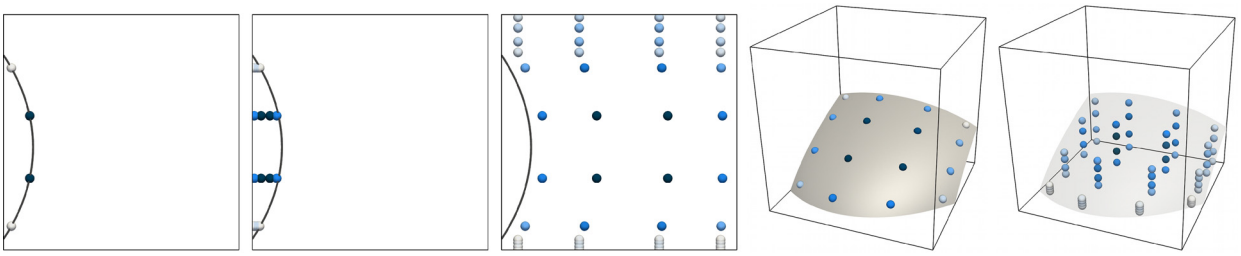

Fig. 3. Examples of quadrature schemes for implicitly defined phase cells, as constructed by the algorithms of [29] with $q = 4$. (*left three*) Integration in two dimensions, for an implicitly defined curve and area on either side. (*right two*) Integration in three dimensions, for an implicitly defined curved surface and the volume underneath. The weights are coloured according to a scale that is normalised for each particular case: ■ denotes a small weight, ■ a medium weight, and ■ a large weight. (Figures adapted from [29].) (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the phase with largest index ("left"), into the phase with smallest index ("right"). Lastly, boundary faces adopt the natural outwards-pointing unit normal, in which "left" means the side of the face interior to the domain.

The notation $[\cdot]$ denotes the jump of a quantity, and is used in two ways: (i) in the definition of PDE problem, to denote the jump of a function across an interface, and (ii) to denote the jump of a piecewise polynomial function across any intraphase or interphase face. In the first case, the jump adopts the same orientation as that of the normal vector of the interface, i.e., on Γ_{ij} we have $[f] := (f_i - f_j)|_{\Gamma_{ij}}$, where f_i is the value of f in phase i. In the second case, the jump adopts the same orientation as the normal of the face. In particular, define u^- to be the trace of the polynomial on the "left" side of the face, u^+ to be the trace of the polynomial on the "right" side of the face, and $[u] = u^- - u^+$. These definitions are also illustrated in Fig. 2.

2.2.2. Quadrature for element and face integration

Of fundamental importance in a finite element method is the weak formulation, requiring integration of functions (most often polynomials) on individual mesh elements and faces. For rectangular elements of the implicitly defined mesh, one can apply standard quadrature schemes, such as tensor product Gauss–Legendre or Gauss–Lobatto quadrature for both the element and its rectangular faces.

For curved elements, curved faces, or flat faces cut by the interface or an implicitly defined domain boundary, the domain of integration is implicitly defined by a level set function. It follows that a quadrature scheme must be computed, and in this work, the high-order accurate quadrature algorithms developed in [29] are adopted. These algorithms have been designed specifically for implicitly defined surfaces and volumes in hyperrectangles, and yield quadrature schemes with the same order of accuracy as tensor-product Gaussian quadrature. In particular, in d dimensions, based on a Gaussian quadrature scheme with q nodes, the number of quadrature nodes is $\mathcal{O}(q^d)$ in the case of volume integrals and $\mathcal{O}(q^{d-1})$ in the case of surface integrals; for sufficiently smooth problems, the order of accuracy is approximately $2q$. Importantly, the quadrature weights computed by these schemes are strictly positive—this ensures that individual elemental mass matrices, which are symmetric positive definite in theory, have the same property in practice. The parameter q is a user-chosen parameter, which is chosen large enough to ensure computed integrals sufficiently minimise associated "variational crimes" within the dG method. Since the integration over a curved element or face implicitly takes into account a Jacobian relating to the curvature of the corresponding domain, it is typically necessary to choose q larger than p. In this work, the parameter is fixed to be $q = 10$ independent of $1 \leq p \leq 5$; this is because the cost of computing the quadrature scheme (accumulated across all curved elements and faces) is essentially negligible compared to other components of the implicit mesh dG framework. The quadrature algorithm is applied to each individual phase cell making up a non-rectangular element; when such an element consists of more than one phase cell, the quadrature schemes are appended and accumulated to define a quadrature scheme for the entire element. Fig. 3 illustrates the typical quadrature node layout and weights computed by the quadrature algorithm with a variety of cases using $q = 4$.

The quadrature algorithms developed in [29] make use of a technique that computes bounds on the range of attainable values of a given function in a given hyperrectangle. This technique is used within the algorithm to identify whether the zero level set of a function intersects the cells of the background quadtree/octree or their boundaries. In other words, the quadrature algorithm automatically identifies "small" phase cells, even in the case, for example, a cell contains an isolated tiny droplet. This feature is exploited in the construction of the implicitly defined mesh to correctly identify the small phase cells in the merging process, thereby avoiding the issues lower-order methods (such as marching cubes or marching tetrahedra) may have in failing to detect such cells (as mentioned in §2.1). However, it is not important to accurately compute the associated volume fractions,[2] and so a smaller value of q can be used for this purpose; in this work, the value $q = 1$ is used.

[2] Recall that the volume fraction quantities are used only to classify non-empty phase cells as either small, large, or entire. To calculate the "true" volume of a mesh element or phase cell, the full-order quadrature scheme can be used.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

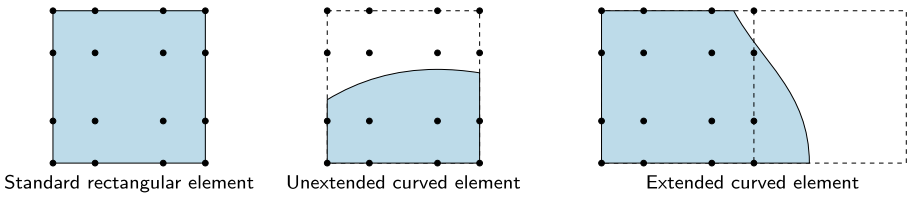

Fig. 4. Nodal basis for the elements of the implicitly defined mesh, demonstrated with $p = 3$, i.e., piecewise bicubic polynomials. (*left*) A standard rectangular element uses the tensor-product Gauss–Lobatto nodes. (*middle*) An unextended curved element, whose shape is implicitly defined by the interface or domain boundary, consisting of only one large phase cell, uses a tensor-product Gauss–Lobatto nodal basis relative to that cell, i.e., its parent cell of the quadtree/octree (dashed). (*right*) Similarly, an extended curved element, which consists of a parent phase cell and one or more child small phase cells, uses a nodal basis relative to its parent cell in the quadtree/octree.

2.2.3. Choice of basis

Although much of a dG method can be stated variationally—and thus independent of a basis—from a computational perspective it is necessary to choose a basis of V_h. In this work a nodal basis is employed, which, for standard rectangular elements, consists of the tensor-product Gauss–Lobatto nodes ($p+1$ nodes per dimension, for a total of $(p+1)^d$ nodes per element), see Fig. 4 left. For simplicity of implementation, curved elements, i.e., both extended and unextended elements (see Fig. 1) adopt the same Gauss–Lobatto nodal basis relative to their "parent" cell in the quadtree/octree. This implies that, in some cases, nodes are "outside" of the element (Fig. 4 middle), whereas in other cases, polynomial values are "extrapolated" based on the nodal values into the extended parts of an element (Fig. 4 right). Nevertheless, two aspects must be emphasised: (i) from a theoretical point of view, this is merely a specification of a basis for V_h; and (ii) with regard to variational statements, L^2 projections, etc., a polynomial on a curved element is determined by the geometry of the element, and not on whether a node is inside or outside. Thus, the viewpoint of "extrapolation" is in some regards simply an interpretation of the role of the nodal basis. Stated differently, since the following dG framework is defined variationally (i.e., via integration over the elements and faces of the mesh), the placement of the nodes is more of a means to an end, rather than being of direct importance like in a finite difference method. On the other hand, this choice of basis does factor into numerical conditioning, such as in the conditioning of elemental mass matrices. In practical settings, for moderately high-order elements ($p \leq 5$), in both two and three dimensions, with a small volume fraction threshold of $\approx 40\%$ (as used in §2.1), experiments indicated that the elemental mass matrices are always sufficiently well-conditioned for double precision arithmetic.

On occasion, it will be convenient to slightly abuse notation and refer to a piecewise polynomial function u in two ways: (i) as a member of V_h, in which case u is to be understood as a function in $L^2(\Omega)$; and (ii) as a vector of coefficients relative to the nodal basis of V_h, in which case $u \in \mathbb{R}^{n_{\text{dof}}}$ where n_{dof} is the number of degrees of freedom. The former is principally used within variational statements, whereas the latter is used in statements involving matrices and linear systems. Given such a u, it should be clear from context which of these interpretations is intended.

2.2.4. Mass matrices

Let M_E denote the mass matrix for an element $E \in \mathcal{E}$ relative to the above choice of basis, i.e.,[3] $M_{E,ij} = \int_E \ell_i \ell_j$, where ℓ_i, $i = 1, \ldots, (p+1)^d$, is an ordering of the $(p+1)^d$ nodal basis functions. In particular, if one adopts the natural ordering given by the position of node $i = (i_1, \ldots, i_d)$ in the tensor-product lattice of Gauss–Lobatto nodes, then it is straightforward to show that, for rectangular elements, M_E is a scalar multiple of a tensor product of one-dimensional mass matrices defined relative to the one-dimensional standard reference element occupying the interval $(-1, 1)$. In this work, these reference mass matrices and their inverses are precomputed to full-precision accuracy and stored in code. For each non-rectangular element, i.e., elements E whose shape is determined implicitly by embedded interfaces or domain boundaries, a quadrature scheme is computed as part of defining the implicit mesh. The scheme is of the form $\int_E f \approx \sum_{k=1}^n w_k f(x_k)$, where x_k are the n quadrature nodes having weights w_k, and is used to define $M_{E,ij} = \sum_{k=1}^n w_k \ell_i(x_k) \ell_j(x_k)$. It can be shown that the resulting discrete mass matrix is automatically symmetric positive definite, based on the operation of the quadrature algorithms in [29] together with the assumption that $q \geq p$. In addition to computing the mass matrix for each non-rectangular element, its Cholesky decomposition is also computed and stored. The Cholesky factorisation of each non-rectangular element is used whenever $M_E^{-1} u$ is calculated for some elemental polynomial u. Last, denote by M the ensemble block-diagonal mass matrix for the entire mesh, i.e., $M = \text{diag}(M_1, \ldots, M_{|\mathcal{E}|})$, where $E_1, \ldots, E_{|\mathcal{E}|}$ is some particular ordering of the $|\mathcal{E}|$ elements of the mesh; note that the ensemble inverse mass matrix M^{-1} is also block-diagonal.

2.2.5. Summary

To summarise the construction so far, given a quadtree/octree of rectangular cells $U_i \subset \mathbb{R}^d$, such that $\Omega \subseteq \bigcup_i U_i$, together with an implicit description of Ω_i for each phase i:

[3] For brevity of presentation, throughout this work, the differential indicating the measure of integration is omitted whenever the context allows it; e.g., $\int_U f = \int_U f(x)\,dx$ whenever $U \subseteq \mathbb{R}^d$ is a volumetric domain, and $\int_U f = \int_U f\,dS$ whenever U is a codimension-one surface.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

1. Compute the volume of each *phase cell* $U_i \cap \Omega_j$ using the quadrature algorithms of [29] with $q = 1$, and classify according to its volume fraction $\theta = |U_i \cap \Omega_j|/|U_i|$: if $\theta = 0$, mark as *empty*; if $0 < \theta < 0.4$, mark as *small*; if $0.4 \leq \theta < 1$, mark as *large*; else mark as *entire*.
2. Merge small phase cells with neighbouring non-small phase cells by assigning to each small phase cell a *parent*: for each small phase cell C, choose from among its neighbours (which are large or entire and in the same phase) sharing a face, edge, or vertex with C (in that order[4]), the cell with largest volume fraction.
3. Define the set of elements as the result of step two, i.e., each element corresponds to either a large or entire phase cell (the element's *parent phase cell*), and zero or more small phase cells (the element's *children phase cells*). The nodal basis of each element is defined to correspond to the tensor-product Gauss–Lobatto nodes positioned relative to its parent phase cell.
4. Compute the geometry of each element: if the element's parent phase cell is entire, and it has no children, then the element is rectangular and a standard reference element can be used. Otherwise, the element is curved: for each of its parent and child phase cells, compute and store a quadrature scheme using the algorithms of [29] with parameter q; with the accumulated quadrature scheme for the entire element, compute its mass matrix and associated Cholesky factorisation.
5. Loop over every face of the mesh by visiting each parent and child phase cell of each element and use the quadtree or octree to find neighbouring phase cells and their associated elements. Most of the faces have an entire phase cell on one side, and are thus rectangular, allowing standard face quadrature rules to be used. For all other faces (e.g., including those on Γ_{ij}), compute and store a quadrature scheme using the quadrature algorithms of [29] with parameter q; in some cases, the quadrature is a curved surface integral applied to the implicitly defined interface Γ_{ij} or perhaps an implicitly defined domain boundary $\partial\Omega$; in other cases, it is applied to a flat face, whose boundary is defined implicitly, which itself is an implicitly defined volume integral but in one fewer dimensions.

The result of this process is a collection of mesh elements and an enumeration of the faces of the mesh. Each curved element and non-rectangular face has a quadrature scheme precomputed for later usage. These quadrature schemes will benefit a variety of operations within the dG framework, including L^2 projections, numerical flux integration, discretisation of advection terms, as well as, for example, computing forces and torques on rigid bodies embedded within a fluid.

2.3. Local discontinuous Galerkin methods for elliptic interface problems

In this section we consider the numerical discretisation of a variety of elliptic interface PDE problems which impose jump conditions on a function and its normal derivative across embedded interfaces. The goal is to develop a discretisation which is symmetric positive (semi)definite, so that one can utilise efficient solution algorithms such as multigrid-preconditioned conjugate gradient methods. In this work, a local discontinuous Galerkin (LDG) [54] approximation is adopted, as the construction offers a variety of properties well-suited to projection operators used within incompressible fluid dynamics.

2.3.1. Model problem description

The simplest elliptic interface problem consists of solving for a function $u : \Omega \to \mathbb{R}$ such that

$$\begin{cases} -\Delta u = f & \text{in } \Omega_i \\ [u] = g_{ij} & \text{on } \Gamma_{ij} \\ [\partial_n u] = h_{ij} & \text{on } \Gamma_{ij} \\ u = g & \text{on } \Gamma_D \\ \partial_n u = h & \text{on } \Gamma_N. \end{cases} \tag{1}$$

Here, $\partial_n := \mathbf{n} \cdot \nabla$ denotes the derivative in the normal direction, while Γ_D and Γ_N denote the components of the domain boundary which impose Dirichlet and Neumann boundary conditions. (Either one of Γ_D or Γ_N may be empty, however $\Gamma_D \cup \Gamma_N = \partial\Omega$ always holds.) Thus, this elliptic interface problem consists of solving for u, with source term f, boundary data g and h, and interfacial jump data g_{ij} and h_{ij}.

2.3.2. Discretisation

In the LDG formulation, a temporary flux variable $\mathbf{q} := \nabla u$ is introduced and the Laplacian is written as the divergence of $\mathbf{q}$. Both $\mathbf{q}$ and its divergence are defined weakly with the assistance of numerical fluxes defined on each mesh face. We define[5] $\mathbf{q} \in V_h^d$ such that

[4] To be more precise, in searching for a suitable candidate neighbour cell for C, neighbouring candidate cells sharing a face are first considered (i.e., extension occurs along only one axis). If no such neighbours exist, then all neighbours that share an edge (in 3D) or vertex (in 2D) are considered (i.e, extension occurs along a diagonal, in two axis directions only). In 3D, if no such neighbours exist, vertex-based neighbours are considered. This ad hoc prioritisation given to the set of neighbours of C minimises the number of axes across which the extension occurs, and improves the aspect ratio of the extended element.

[5] See the later footnote regarding equation (4).

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

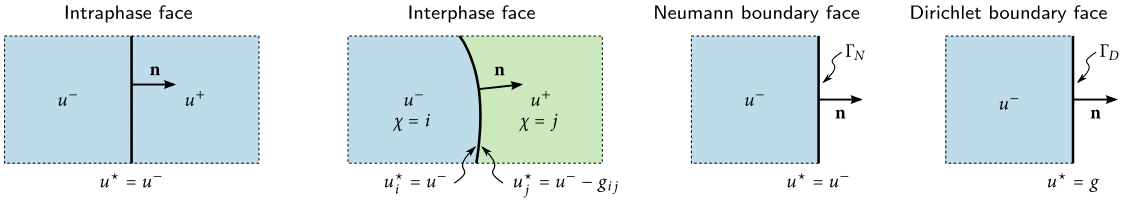

Fig. 5. Schematic of the numerical flux function $u^\star$ defined by (3) and used within the evaluation of the discrete gradient of u. Except for interphase faces, the flux is single-valued and defined by the polynomial on the "left" of the face (recall that the normal $\mathbf{n}$ points "left-to-right"). For interphase faces, in order to compensate for imposed jump conditions $[u] = g_{ij}$ on Γ_{ij} $(i > j)$, the flux depends on the phase of the element.

$$\int_E \mathbf{q} \cdot \boldsymbol{\omega} = -\int_E u \nabla \cdot \boldsymbol{\omega} + \int_{\partial E} u^\star_{\chi(E)} \boldsymbol{\omega} \cdot \mathbf{n} \tag{2}$$

for every element $E \in \mathcal{E}$ having phase $\chi(E)$ and every test function $\boldsymbol{\omega} \in V_h^d$. There are a variety of methods for defining the numerical flux $u^\star$ leading to different discretisations. In this work, a one-sided directional principle is employed, in which $u^\star$ is defined by elements on the "left" of the face (see also [55,56]). Moreover, and although the numerical flux is traditionally single-valued, in this application involving interfacial jump conditions it is advantageous to define a multi-valued numerical flux. Referring to Fig. 5, define

$$u^\star_\chi := \begin{cases} u^- & \text{on any intraphase face,} \\ u^- & \text{on } \Gamma_{\chi i} \text{ if } \chi > i, \\ u^- - g_{i\chi} & \text{on } \Gamma_{i\chi} \text{ if } i > \chi, \\ u^- & \text{on } \Gamma_N, \\ g & \text{on } \Gamma_D. \end{cases} \tag{3}$$

Note that the flux is multivalued on interphase faces with value depending on χ, i.e., the phase of the element in consideration in (2). On these faces, the interfacial jump condition data is taken into account as follows: if an element "reaches across" the interface into another phase in order to determine its numerical flux, a jump in the solution occurs—this jump is then accounted for by subtracting the interfacial jump condition $[u] = g_{ij}$. Meanwhile, for a face on the Neumann part of the domain boundary, the numerical flux is given by the interior element; for a face on the Dirichlet part of the domain boundary, the numerical flux is given by the boundary data.

Note that (2) is equivalent[6] to

$$\int_E \mathbf{q} \cdot \boldsymbol{\omega} = \int_E \nabla u \cdot \boldsymbol{\omega} + \int_{\partial E} (u^\star_{\chi(E)} - u) \boldsymbol{\omega} \cdot \mathbf{n}, \tag{4}$$

which upon summing over every element of the mesh states that, for any $\boldsymbol{\omega} \in V_h^d$,

$$\begin{aligned} \int_\Omega \mathbf{q} \cdot \boldsymbol{\omega} &= \sum_{E \in \mathcal{E}} \int_E \nabla u \cdot \boldsymbol{\omega} + \int_{\Gamma_0} (u^\star - u^-) \boldsymbol{\omega}^- \cdot \mathbf{n} - (u^\star - u^+) \boldsymbol{\omega}^+ \cdot \mathbf{n} + \sum_{i>j} \int_{\Gamma_{ij}} (u_i^\star - u^-) \boldsymbol{\omega}^- \cdot \mathbf{n} - (u_j^\star - u^+) \boldsymbol{\omega}^+ \cdot \mathbf{n} \\ &\quad + \int_{\Gamma_D} (u^\star - u^-) \boldsymbol{\omega}^- \cdot \mathbf{n} + \int_{\Gamma_N} (u^\star - u^-) \boldsymbol{\omega}^- \cdot \mathbf{n} \\ &= \sum_i \int_{E_i} \nabla u \cdot \boldsymbol{\omega} - \int_{\Gamma_0} (u^- - u^+) \boldsymbol{\omega}^+ \cdot \mathbf{n} - \sum_{i>j} \int_{\Gamma_{ij}} (u^- - g_{ij} - u^+) \boldsymbol{\omega}^+ \cdot \mathbf{n} + \int_{\Gamma_D} (g - u^-) \boldsymbol{\omega}^- \cdot \mathbf{n}. \end{aligned} \tag{5}$$

We now define the following operators:

- Let $\nabla_h : V_h \to V_h^d$ be the broken gradient operator, and let $L : V_h \to V_h^d$ be the lifting operator, such that

[6] On a first reading, (2) and (4) may be regarded as being equivalent for motivational purposes. On closer inspection, one may note that there is a subtle distinction between (2) and (4) when approximate numerical quadrature schemes are utilised (e.g., as used on a curved element E), as such schemes do not in general exactly satisfy the identity of integration by parts. In other words, the numerical approximation of (2) may result in a different polynomial for $\mathbf{q}$ on E compared to the numerical approximation of (4); the difference corresponds to the truncation error of the quadrature scheme. In the nomenclature of [28], (2) is referred to as the *weak form* and (4) as the *strong form*. The difference between the two forms comes into play in the symmetry of the final discrete Laplacian operator, as $\mathbf{q}$ is defined via the strong form and the divergence of $\mathbf{q}$ via the weak form. In short, in the remainder of this section, $\mathbf{q}$ is to be interpreted as being defined solely by (4) whether or not approximate numerical quadrature is used.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

$$\int_\Omega (\nabla_h u)\cdot\boldsymbol{\omega} = \sum_{E\in\mathcal{E}}\int_E \nabla u\cdot\boldsymbol{\omega},$$
$$\int_\Omega (Lu)\cdot\boldsymbol{\omega} = \int_{\Gamma_0}(u^+-u^-)\boldsymbol{\omega}^+\cdot\mathbf{n} + \sum_{i>j}\int_{\Gamma_{ij}}(u^+-u^-)\boldsymbol{\omega}^+\cdot\mathbf{n} - \int_{\Gamma_D} u^-\boldsymbol{\omega}^-\cdot\mathbf{n},$$

holds for every $\boldsymbol{\omega}\in V_h^d$. Thus, ∇_h evaluates the gradient of each polynomial on each element, while L computes the jump in u across the set of faces (excluding those on Γ_N), multiplies by the associated normal vector of each face, and "lifts" the result into the interior. Note that these operators are uniquely defined, since, if $\int_\Omega \tilde{\boldsymbol{\omega}}\cdot\boldsymbol{\omega}=0$ for all $\boldsymbol{\omega}\in V_h^d$, then $\tilde{\boldsymbol{\omega}}$ must necessarily be zero.

- Define $J_g(g,g_{ij})\in V_h^d$ such that

$$\int_\Omega J_g(g,g_{ij})\cdot\boldsymbol{\omega} = \sum_{i>j}\int_{\Gamma_{ij}} g_{ij}\,\boldsymbol{\omega}^+\cdot\mathbf{n} + \int_{\Gamma_D} g\,\boldsymbol{\omega}^-\cdot\mathbf{n},$$

holds for every $\boldsymbol{\omega}\in V_h^d$. Note that the domain of the operator J_g is not specified, since we do not assume the interfacial jump data g_{ij} or boundary data g belongs to any particular space. (In some applications however, these data may in fact be the trace of a function in V_h.)

With these definitions, (5) is equivalent to the statement that

$$\mathbf{q} = (\nabla_h + L)u + J_g(g,g_{ij}) = Gu + J_g(g,g_{ij}), \tag{6}$$

where $G: V_h\to V_h^d$ is the *discrete gradient operator*,

$$G := \nabla_h + L. \tag{7}$$

The formula (6) implements the weak statement that $\mathbf{q}=\nabla u$, taking into account the interfacial jump conditions and Dirichlet boundary data.

We now consider the weak formulation for computing the divergence of $\mathbf{q}$. Given $\mathbf{q}\in V_h^d$, define $w\in V_h$ such that

$$\int_E wv = -\int_E \mathbf{q}\cdot\nabla v + \int_{\partial E} v\,\mathbf{q}^\star_{\chi(E)}\cdot\mathbf{n} \tag{8}$$

holds for every test function $v\in V_h$ and every element $E\in\mathcal{E}$ with phase $\chi(E)$. In this case, the numerical flux is again determined by a one-sided directional principle, but in the opposite direction to that used by $u^\star$. For simplicity of presentation, the numerical flux in the following is vector-valued; however, only the normal component of the flux is used. We define

$$\mathbf{q}^\star_\chi := \begin{cases} \mathbf{q}^+ & \text{on any intraphase face,}\\ \mathbf{q}^+ + h_{\chi i}\mathbf{n} & \text{on } \Gamma_{\chi i} \text{ if } \chi > i,\\ \mathbf{q}^+ & \text{on } \Gamma_{i\chi} \text{ if } i > \chi,\\ \mathbf{q}^- & \text{on } \Gamma_D,\\ h\mathbf{n} & \text{on } \Gamma_N, \end{cases} \tag{9}$$

with the orientation of the normal $\mathbf{n}$ adopted from the associated interface/boundary. Thus, the numerical flux for the divergence of $\mathbf{q}$ is defined by elements on the "right". It is multivalued on interphase faces, where the interfacial jump condition is taken into account whenever an element reaches across the interface into another phase. For a face on the Neumann part of the domain boundary, the flux is given by the boundary data; for the Dirichlet part of the domain boundary, it is given by the interior element. Summing (8) over every element, we have that, for every $v\in V_h$,

$$\begin{aligned}\int_\Omega wv &= -\sum_{E\in\mathcal{E}}\int_E \mathbf{q}\cdot\nabla v + \int_{\Gamma_0}(v^- - v^+)\mathbf{q}^\star\cdot\mathbf{n} + \sum_{i>j}\int_{\Gamma_{ij}}(v^-\mathbf{q}_i^\star - v^+\mathbf{q}_j^\star)\cdot\mathbf{n} + \int_{\Gamma_D} v^-\mathbf{q}^\star\cdot\mathbf{n} + \int_{\Gamma_N} v^-\mathbf{q}^\star\cdot\mathbf{n}\\ &= -\sum_{E\in\mathcal{E}}\int_E \mathbf{q}\cdot\nabla v + \int_{\Gamma_0}(v^- - v^+)\mathbf{q}^+\cdot\mathbf{n} + \sum_{i>j}\int_{\Gamma_{ij}} v^- h_{ij} + (v^- - v^+)\mathbf{q}^+\cdot\mathbf{n} + \int_{\Gamma_D} v^-\mathbf{q}^-\cdot\mathbf{n} + \int_{\Gamma_N} v^- h.\end{aligned} \tag{10}$$

Similar to the operator J_g defined above, let $J_h(h,h_{ij})\in V_h$ be such that $\int_\Omega J_h(h,h_{ij})\,v = \sum_{i>j}\int_{\Gamma_{ij}} h_{ij}\,v^- + \int_{\Gamma_N} h\,v^-$ for all $v\in V_h$. Then, (10) is equivalent to the statement that, for all $v\in V_h$,

$$(w,v) = -(\nabla_h v,\mathbf{q}) - (Lv,\mathbf{q}) + \big(J_h(h,h_{ij}),v\big) = -(Gv,\mathbf{q}) + \big(J_h(h,h_{ij}),v\big). \tag{11}$$

This is the weak statement for $w=\nabla\cdot\mathbf{q}$, taking into account the interfacial jump conditions and Neumann boundary data.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

We are almost in a position to define the discrete solution of the elliptic interface problem (1). Formally, it consists of finding $u \in V_h$ such that $\mathbf{q} = \nabla u$, $w = \nabla \cdot \mathbf{q}$, and $-w = f$ (all in the weak sense), which, upon combining (6) and (11), implies that u solves the variational problem

$$(Gu, Gv) + \big(Gv,\, J_g(g, g_{ij})\big) - \big(J_h(h, h_{ij}), v\big) = (f, v) \quad \text{for all } v \in V_h.$$

However, as is commonplace in discontinuous Galerkin methods, it is sometimes necessary to include penalty terms to ensure well-posedness of the discrete problem [28,54]. These stabilisation parameters, which couple elements together via shared faces by penalising discontinuities, are used to ensure that the kernel of the discrete Laplacian is trivial when Γ_D is non-empty, or contains only the constant functions when Γ_D is empty. To implement stabilisation, we consider three types of penalty parameters: $\tau_0 \geq 0$, $\tau_{ij} \geq 0$ and $\tau_D \geq 0$, for the intraphase, interphase and Dirichlet boundary mesh faces, respectively. For intraphase faces, discontinuities in the solution u are penalised by adding to the variational problem a term of the form $\tau_0 \int_{\Gamma_0} [u][v]$. For the Dirichlet part of the domain boundary, the difference between u and g is penalised by adding a term of the form $\tau_D \int_{\Gamma_D} (u^- - g)v^-$. Last, for the interphase faces, the jump in u (after adjusting for interfacial jump conditions) is penalised by adding a term of the form $\tau_{ij} \int_{\Gamma_{ij}} ([u] - g_{ij})[v]$. (For motivation regarding the particular forms employed here the reader is referred to, e.g., [28,54] for discussion.) With these additions to the variational problem, we are led to the final variational statement for solving the elliptic interface problem (1): find $u \in V_h$ such that

$$(Gu, Gv) + \big(Gv,\, J_g(g, g_{ij})\big) - \big(J_h(h, h_{ij}), v\big) + \tau_0 \int_{\Gamma_0} [u][v] + \sum_{i>j} \tau_{ij} \int_{\Gamma_{ij}} ([u] - g_{ij})[v] + \tau_D \int_{\Gamma_D} (u^- - g)v^- = (f, v) \tag{12}$$

holds for every $v \in V_h$. Several remarks are in order.

- Note that the bilinear part of the variational form, i.e., $a : V_h \times V_h \to \mathbb{R}$, where

$$a(u, v) := (Gu, Gv) + \tau_0 \int_{\Gamma_0} [u][v] + \sum_{i>j} \tau_{ij} \int_{\Gamma_{ij}} [u][v] + \tau_D \int_{\Gamma_D} u^- v^-, \tag{13}$$

  is symmetric positive semidefinite. In fact, assuming that all of the penalty parameters are positive (and the domain simply connected), if u is such that $a(u, u) = 0$, then u must be uniformly constant throughout Ω. A simple proof is as follows: suppose $a(u, u) = 0$; then all of the penalty integrals are zero, and so u is continuous across all interphase and intraphase faces; thus u is continuous throughout Ω; hence, since $\|Gu\|^2 = 0$ and $Lu = 0$, it follows that $\|\nabla_h u\|^2 = 0$; therefore, u is constant throughout Ω. If, in addition, Γ_D is non-empty, then the last penalty term of (13) implies that $u = 0$, showing that in this case a is in fact symmetric positive definite.
- It follows from the previous point that the discrete problem is well-posed: provided the penalty parameters are suitably chosen, if Γ_D is non-empty, then there exists a unique solution $u \in V_h$ to (12). Alternatively, if Γ_D is empty, i.e., only Neumann boundary conditions are imposed, then, assuming f and the interfacial jump data satisfy the compatibility condition $(f + J_h(h, h_{ij}), 1) = 0$, then $u \in V_h$ is unique up to a global constant.
- Note that, putting aside the penalty terms, the bilinear component of the variational form is simply (Gu, Gv). Therefore the discrete Laplacian has the form $-G^*G$, where G^* is the adjoint of G. In other words, the negative adjoint of G plays the role of a discrete divergence operator. This property is also reflective of the choice of the one-sided directional strategy defining the numerical fluxes $u^\star$ and $\mathbf{q}^\star$—by "upwinding" from the left to calculate the gradient, the adjoint of the gradient should "upwind" from the right; the composition of the gradient and divergence therefore makes use of information in both directions.
- To illustrate, consider a one-dimensional case in which the polynomial space is piecewise constant and denote the solution by u_i on each interval $[ih, (i+1)h]$ (where h is the element width). Then the gradient operator upwinds from the left and yields $(\nabla u)_i = \frac{1}{h}(u_i - u_{i-1})$; the divergence operator upwinds from the right and yields $(\nabla \cdot q)_i = \frac{1}{h}(q_{i+1} - q_i)$. The combined operation yields $\nabla \cdot \nabla u = \frac{1}{h^2}(u_{i+1} - 2u_i + u_{i-1})$, which is the standard second-order finite difference operator. This extends to higher-order elements, and in one dimension yields a stencil for the discrete Laplacian of width ± 1. On a regular Cartesian grid in 2D (respectively, 3D) the discrete Laplacian is a stencil involving 5 (respectively, 7) elements in the standard cross pattern, also of width ± 1. For more general unstructured meshes, as for example in the implicit meshes used in this work, the stencil may be less compact and involve elements in a neighbourhood two elements away in connectivity.
- As will be demonstrated in forthcoming convergence tests, experiments indicate that the above LDG formulation has optimal-order accuracy: the computed solution is order $p + 1$ accurate in the L^∞ norm, assuming the interfacial jump, boundary, and source data admit a sufficiently smooth solution. This applies to curved domains as well as multiple interfaces. The discrete operators are also amenable to geometric multigrid methods, as will be discussed in §2.4.
- Regarding the penalty parameters, strictly positive parameters are sufficient to ensure invertibility of the discrete problem. However this is not a necessary condition. For example, on a regular Cartesian grid (without any interfaces), τ_0 can be taken zero. For meshes with less structure on element connectivity, such as a triangulated mesh, or the implicit

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

meshes considered in this work, a positive τ_0 may be necessary in order to ensure there are no non-trivial components of the element connectivity graph associated with the discrete Laplacian [28,54]. On the other hand, excessively large penalty parameters tend to worsen the conditioning of the linear system, so it is advantageous to keep them a moderate size. It is outside the scope of this work to extensively analyse the effect of the penalty parameters on accuracy and conditioning. Instead, the following is noted, reflective of a wide range of tests involving the multigrid-preconditioned conjugate gradient algorithm described in §2.4: it was found that $\tau_{ij} \approx 0.1$ was effective in ensuring high-order accuracy and excellent conditioning, especially in terms of the performance of multigrid cycles; $\tau_0 = 0$ sufficed in all cases, despite the aforementioned warnings; and $\tau_D \approx 1000$ allows multigrid relaxation to efficiently impose Dirichlet boundary conditions, without causing any unnecessary ill-conditioning.

- Last, it is worthwhile to note that the choice of numerical flux for $u^\star$ and $\mathbf{q}^\star$ can be generalised in various ways. One such possibility is to replace the numerical flux on interphase faces with a general convex combination of the left and right values, i.e., to choose

$$u^\star_\chi = \begin{cases} c_1 u^- + c_2(u^+ + g_{\chi i}) & \text{on } \Gamma_{\chi i} \text{ if } \chi > i, \\ c_1(u^- - g_{i\chi}) + c_2 u^+ & \text{on } \Gamma_{i\chi} \text{ if } i > \chi, \end{cases}$$

and correspondingly weight the numerical flux for $\mathbf{q}$ in the opposite direction,

$$\mathbf{q}^\star_\chi = \begin{cases} c_2 \mathbf{q}^- + c_1(\mathbf{q}^+ + h_{\chi i}\mathbf{n}) & \text{on } \Gamma_{\chi i} \text{ if } \chi > i, \\ c_2(\mathbf{q}^- - h_{i\chi}\mathbf{n}) + c_1 \mathbf{q}^+ & \text{on } \Gamma_{i\chi} \text{ if } i > \chi. \end{cases}$$

The choice $c_1 = 1, c_2 = 0$ corresponds to the one-sided strategy defined above in (3) and (9), while $c_1 = 0, c_2 = 1$ "upwinds" in the opposite direction, and $c_1 = c_2 = \frac{1}{2}$ is a central flux. (Note that these fluxes appropriately compensate for the imposed interfacial jump conditions $[u] = g_{ij}$ and $[\mathbf{q} \cdot \mathbf{n}] = h_{ij}$ on Γ_{ij}.) According to preliminary experiments, the performance of multigrid solvers for elliptic interface problems with very large jumps in ellipticity coefficient (see also §2.3.4) can be considerably improved by carefully choosing the weights c_1 and c_2 in a way that depends on the interfacial values of the ellipticity coefficient. Such a procedure may therefore lead to large practical benefits for multiphase fluid flow, particularly when there is a large ratio in the density of the fluids involved, however we do not further pursue this idea in the present work.

2.3.3. Briefly on numerical implementation

With the aid of the quadrature schemes discussed in §2.2.2, one can precompute the lifting operator L and therefore the discrete gradient operator G defined in (7) as a block-sparse matrix (with each block corresponding to a particular element) such that a block (i, j) is non-zero if there is (at least one) face between element i and element j with j on the "left" of i. In particular, recalling that the codomain of G is the set of piecewise polynomial vector fields, denote by G_k, $1 \le k \le d$, as the components of the vector field, such that $Gu = (G_1 u, \ldots, G_d u)$. Then the bilinear form (Gu, Gv) can be rewritten as $v^\intercal(\sum_{k=1}^d G_k^\intercal M G_k)u$, where M is the block-diagonal mass matrix. (Here and in the following, we slightly abuse notation by considering u and v as elements of V_h as well as vectors of coefficients relative to the nodal basis of V_h.) A similar consideration holds for the jump penalty integrals, allowing (12) to be rewritten as

$$\begin{aligned} v^\intercal\Big(\sum_{1\le k\le d} G_k^\intercal M G_k\Big)u + \tau_0 v^\intercal A_0 u + \sum_{i>j} \tau_{ij} v^\intercal A_{ij} u + \tau_D v^\intercal A_D u \\ = v^\intercal M \tilde{f} - v^\intercal\Big(\sum_{1\le k\le d} G_k^\intercal M J_{g,k}(g, g_{ij})\Big) + v^\intercal M J_h(h, h_{ij}) + \sum_{i>j} \tau_{ij} v^\intercal a_{ij}(g_{ij}) + \tau_D v^\intercal a_D(g), \end{aligned} \tag{14}$$

where

- A_0 is the block-sparse symmetric matrix such that $v^\intercal A_0 u = \int_{\Gamma_0} [u][v]$ for all u, v. Block (i, j) of A_0 is nonzero if and only if there exists an intraphase face between element i and element j.
- A_{ij} is the block-sparse symmetric matrix such that $v^\intercal A_{ij} u = \int_{\Gamma_{ij}} [u][v]$, in which block (i, j) is nonzero if and only if element i and j share an interphase face.
- A_D is such that $v^\intercal A_D u = \int_{\Gamma_D} [u][v]$; it is block-diagonal and symmetric, having non-zero diagonal blocks only for elements situated on the Dirichlet boundary.
- $a_{ij}(g_{ij})$ and $a_D(g)$ are the (unique) elements of V_h which lift the jump data g into the interior, i.e., $v^\intercal a_{ij}(g_{ij}) = \int_{\Gamma_{ij}} g_{ij}[v]$ and $v^\intercal a_D(g) = \int_{\Gamma_D} g v^-$ for all $v \in V_h$. Similarly, $\tilde{f}$ is such that $v^\intercal M \tilde{f} = (v, f)$ for all $v \in V_h$, i.e., $\tilde{f}$ is the L^2 projection of f; alternatively, one can often replace the projection with an approximation in which $\tilde{f}$ is defined as the nodal interpolant of f.

Since (14) holds for every v, it follows that

$$\Big[\Big(\sum_{1\le k\le d} G_k^\intercal M G_k\Big) + \tau_0 A_0 + \sum_{i>j}\tau_{ij}A_{ij} + \tau_D A_D\Big]u$$
$$= M\tilde{f} - \Big(\sum_{1\le k\le d} G_k^\intercal M J_{g,k}(g, g_{ij})\Big) + M J_h(h, h_{ij}) + \sum_{i>j}\tau_{ij}a_{ij}(g_{ij}) + \tau_D a_D(g). \tag{15}$$

The matrix acting on u in (15) is block-sparse, symmetric, and positive semidefinite. (It is in fact positive definite if Γ_D is nonempty and the penalty parameters are suitably chosen.) In this work, the matrix is precomputed in a block-sparse format; doing so allows for a simpler implementation of multigrid and conjugate gradient algorithms. Note also that, pre-multiplying (15) by M^{-1} and putting aside the penalty terms for the sake of illustration, one obtains the linear system $M^{-1}\sum_k G_k^\intercal M G_k u = \tilde{f} + \cdots$; thus, the matrix $M^{-1}\sum_k G_k^\intercal M G_k$ essentially plays the role of the discrete Laplacian. This interpretation factors into the design of multigrid algorithms, as discussed in §2.4.

2.3.4. Extension to variable coefficient elliptic operators

The previous elliptic interface problem considered the case in which the ellipticity coefficient is uniformly constant. A more general elliptic interface problem involves a variable coefficient elliptic operator,

$$\begin{cases} -\nabla\cdot(\alpha_i\nabla u) = f & \text{in } \Omega_i \\ [u] = g_{ij} & \text{on } \Gamma_{ij} \\ \mathbf{n}\cdot[\alpha\nabla u] = h_{ij} & \text{on } \Gamma_{ij} \\ u = g & \text{on } \Gamma_D \\ \alpha\partial_n u = h & \text{on } \Gamma_N, \end{cases} \tag{16}$$

where α_i is phase-dependent coefficient, either a scalar or a matrix, which may vary in space. In particular, the problem is non-degenerate if α_i is either a positive scalar or a positive-definite matrix. It arises, for example, in multiphase fluid flow problems involving phase-dependent fluid density as well as in multi-scale modelling and multi-physics problems where α could relate to material phase changes like evaporation.

It is relatively straightforward to adapt the above LDG formulation to this new problem involving variable coefficients. Define $\tilde{\mathbf{q}} \in V_h^d$ via (6) as the weak gradient of u, taking into account the interfacial jump condition $[u] = g_{ij}$ on Γ_{ij}. Next, formally define $\mathbf{q} := \alpha\tilde{\mathbf{q}}$. Note, however, that $\mathbf{q}$ may not be an element of V_h^d, since α may vary in space. Nevertheless, one can follow the construction as before, taking the weak divergence of $\mathbf{q}$ and compensating for the interfacial jump condition $\mathbf{n}\cdot[\alpha\nabla u] = h_{ij}$ with unaltered numerical flux (9). This results in the following variational problem: find $u \in V_h$ such that

$$(\alpha G u, G v) + \big(G v, \alpha J_g(g, g_{ij})\big) - \big(J_h(h, h_{ij}), v\big) + \tau_0\int_{\Gamma_0}[u][v] + \sum_{i>j}\tau_{ij}\int_{\Gamma_{ij}}([u]-g_{ij})[v] + \tau_D\int_{\Gamma_D}(u^- - g)v^- = (f, v) \tag{17}$$

holds for every $v \in V_h$. Note that the essential component of the bilinear form, $(\alpha G u, G v)$ continues to be symmetric positive semidefinite, since α is assumed to be symmetric positive definite.

- In the present work, we only have need for the simplest case in which $\alpha_i > 0$ is scalar, uniformly constant throughout Ω_i. In this case, $\mathbf{q} \in V_h^d$ and only two simple modifications of the linear system (15) are necessary: letting α_n denote the nodal interpolant of α (i.e., $v^\intercal M\alpha_n = \int_\Omega \alpha v$ for all $v \in V_h$), then $\sum_{k=1}^d G_k^\intercal M G_k$ in (15) is replaced by $\sum_{k=1}^d G_k^\intercal M\mathcal{A} G_k$ where $\mathcal{A} = \mathrm{diag}(\alpha_n)$, while $\sum_{k=1}^d G_k^\intercal M J_{g,k}(g, g_{ij})$ is replaced by $\sum_{k=1}^d G_k^\intercal M\mathcal{A} J_{g,k}(g, g_{ij})$.
- For the more general case, in which α is a symmetric positive definite matrix varying in space, note that the inner products in (17) depending on α are no longer integrals of the product of two functions in V_h. Consequently, one must take care in computing the integrals to avoid or minimise associated "variational crimes," as it may be necessary to use higher-order quadrature schemes. One may also need to consider rapidly varying coefficients of ellipticity, such as in material phase change problems like solidification and evaporation in which α may contain boundary layers. Alternatively, provided α is sufficiently smooth, a nodal interpolant may suffice; in this case, to maintain symmetry of the matrix in the resulting linear system, one possibility is to use the nodal interpolant of $\sqrt{\alpha}$ (i.e., the principal square root of α) such that the primary component of the linear system reads $\sum_{k=1}^d G_k^\intercal\mathcal{A}M\mathcal{A}G_k$ where $\mathcal{A} = \mathrm{diag}(\sqrt{\alpha})$. Together with a suitable implementation of $(Gv, \alpha J_g(g, g_{ij}))$, this discretisation of (16) results in a symmetric positive (semi)definite linear system to solve for u.
- Note that, in addition to the dependence on element size h, polynomial degree p and choice of penalty parameters τ [28,54], the conditioning of the elliptic interface problem in (16) also depends on α. Although it is difficult to fully characterise the conditioning for general α, one particular observation is that large jumps in α across the interface create gaps in the spectrum of the linear operator and consequently increases the condition number in proportion to the jump ratio. This may impact the performance of associated linear solvers. As shown in §2.4, it is possible to design multigrid solvers which mitigate these effects by designing coarse mesh spaces which sharply represent the interface and the discontinuities in α.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

2.3.5. Extension to a vector-valued elliptic interface problem

We will also need to solve the following vector-valued elliptic interface problem: find $\mathbf{u} : \Omega \to \mathbb{R}^d$ such that

$$\begin{cases} -\nabla \cdot (\alpha_i(\nabla \mathbf{u} + \nabla \mathbf{u}^\mathsf{T})) = \mathbf{f} & \text{in } \Omega_i \\ [\mathbf{u}] = \mathbf{g}_{ij} & \text{on } \Gamma_{ij} \\ [\alpha(\nabla \mathbf{u} + \nabla \mathbf{u}^\mathsf{T})] \cdot \mathbf{n} = \mathbf{h}_{ij} & \text{on } \Gamma_{ij} \\ \mathbf{u} = \mathbf{g} & \text{on } \Gamma_D \\ (\alpha \nabla \mathbf{u}) \cdot \mathbf{n} = \mathbf{h} & \text{on } \Gamma_N. \end{cases} \tag{18}$$

In this case, the source terms and boundary/jump data are vector-valued functions, as is the solution $\mathbf{u}$. (We assume here that $\alpha_i > 0$ is constant on each phase Ω_i, though extension to variable α or symmetric positive definite-valued α is straightforward.) The elliptic operator in this problem arises from the divergence of the viscous stress for a fluid with velocity field $\mathbf{u}$; it will have application for multiphase fluid flow in which different phases have different viscosities μ, as well as in free surface flow.

Although the problem in (18) does not depend on the coordinate system, it is simplest to derive a discrete form assuming a Cartesian coordinate system. Let $u_k \in V_h$ be the kth component of the vector field $\mathbf{u} \in V_h^d$. Define $\mathbf{q}_k = G u_k$ to be the weak gradient of u_k, taking into account the interfacial jump condition $[u_k] = \mathbf{g}_{ij} \cdot \mathbf{e}_k$ via the straightforward analogy of (6). Let the components of $\mathbf{q}_k$ be denoted $q_{k,\ell}$, $\ell = 1, \ldots, d$. Furthermore, define the components of the symmetric stress tensor $\boldsymbol{\sigma} := \alpha(\nabla \mathbf{u} + \nabla \mathbf{u}^\mathsf{T})$ as $\sigma_{k,\ell} = \alpha(q_{k,\ell} + q_{\ell,k})$. Considering each row $\boldsymbol{\sigma}_k$ of $\boldsymbol{\sigma}$ as an element of V_h^d, we take the weak divergence of $\boldsymbol{\sigma}_k$, taking into account its interfacial jump condition $[\boldsymbol{\sigma}_k] = \mathbf{h}_{ij} \cdot \mathbf{e}_k$, and set the result equal to $\mathbf{f} \cdot \mathbf{e}_k$. This leads to the following linear system for $\mathbf{u} \in V_h^d$: solve for $u_1, \ldots, u_d \in V_h$ such that, for each $1 \le \ell \le d$,

$$\begin{aligned} &\sum_{1 \le k \le d} G_k^\mathsf{T} M \mathcal{A} (G_k u_\ell + G_\ell u_k) + \tau_0 A_0 u_\ell + \sum_{i>j} \tau_{ij} A_{ij} u_\ell + \tau_D A_D u_\ell \\ &\qquad = M \mathbf{f} \cdot \mathbf{e}_\ell - \Big(\sum_{1 \le k \le d} G_k^\mathsf{T} M \mathcal{A} J_{k\ell}(\mathbf{g}, \mathbf{g}_{ij}) \Big) + M J_h(\mathbf{h} \cdot \mathbf{e}_\ell, \mathbf{h}_{ij} \cdot \mathbf{e}_\ell) + \sum_{i>j} \tau_{ij} a_{ij}(\mathbf{g}_{ij} \cdot \mathbf{e}_l) + \tau_D a_D(\mathbf{g} \cdot \mathbf{e}_l), \end{aligned} \tag{19}$$

where $\mathcal{A}$ is the diagonal matrix such that $\mathcal{A}u = \alpha u$ and $J_{k\ell} = J_{g,k}(\mathbf{g} \cdot \mathbf{e}_\ell, \mathbf{g}_{ij} \cdot \mathbf{e}_\ell) + J_{g,\ell}(\mathbf{g} \cdot \mathbf{e}_k, \mathbf{g}_{ij} \cdot \mathbf{e}_k)$ is the source term depending on the Dirichlet boundary and jump data, appropriately symmetrised as per the definition of $\boldsymbol{\sigma}$. Note that, as one may expect from (18), the components of the vector field $\mathbf{u}$ are coupled. To illustrate, in three dimensions, and ignoring the penalty terms, the linear system in (19) has block form

$$\begin{pmatrix} 2G_1^\mathsf{T} M \mathcal{A} G_1 + G_2^\mathsf{T} M \mathcal{A} G_2 + G_3^\mathsf{T} M \mathcal{A} G_3 & G_2^\mathsf{T} M \mathcal{A} G_1 & G_3^\mathsf{T} M \mathcal{A} G_1 \\ G_1^\mathsf{T} M \mathcal{A} G_2 & G_1^\mathsf{T} M \mathcal{A} G_1 + 2G_2^\mathsf{T} M \mathcal{A} G_2 + G_3^\mathsf{T} M \mathcal{A} G_3 & G_3^\mathsf{T} M \mathcal{A} G_2 \\ G_1^\mathsf{T} M \mathcal{A} G_3 & G_2^\mathsf{T} M \mathcal{A} G_3 & G_1^\mathsf{T} M \mathcal{A} G_1 + G_2^\mathsf{T} M \mathcal{A} G_2 + 2G_3^\mathsf{T} M \mathcal{A} G_3 \end{pmatrix} \times \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} \cdots \\ \cdots \\ \cdots \end{pmatrix},$$

while the penalty matrices add a diagonal component to this block form. Thus, the solution $\mathbf{u} \in V_h^d$ depends on and couples across all components of the interfacial jump data and boundary data. Moreover, the matrix of (19) is symmetric positive semidefinite (in fact, it is positive definite if Γ_D is nonempty and the penalty parameters are suitably chosen), and is amenable to both multigrid and conjugate gradient methods.

2.3.6. Convergence tests

Thus far, we have considered a local dG discretisation of various elliptic interface problems, suitable for implicitly defined meshes and multiphase domains. Each of these methods yield optimal-order accuracy, and this property is demonstrated here with a series of grid convergence tests. Three problem types are considered:

(a) *Uniform ellipticity coefficient on a curved, four-phase domain and interfaces with triple junctions.* This test problem consists of solving (1) on the two-dimensional disc $\Omega = \{x^2 + y^2 < 1\}$, with Neumann boundary conditions, $\Gamma_N = \partial\Omega$, $\Gamma_D = \emptyset$, with a "H" shaped interface defined by a four-phase system, as depicted in Fig. 6. The interfacial jump data, boundary data, and right hand side source terms in (1) are defined such that the exact solution is given by $u = \cos x \sin y$, $u = x^2 + y^2$, $u = \sin x \sin y$, and $u = \frac{1}{4}e^{x+y} - 1$, for phases A, B, C, and D, respectively.

(b) *Two-phase elliptic interface problem, with a large ratio jump in ellipticity coefficient.* This test problem consists of solving (16), in two and three dimensions:
- In two dimensions, the chosen domain is a square, $\Omega = (-\frac{1}{2}, \frac{1}{2}) \times (-\frac{1}{2}, \frac{1}{2})$, Dirichlet boundary conditions are imposed on the bottom face, $\Gamma_D = \{y = -\frac{1}{2}\}$, Neumann boundary conditions on the top face, $\Gamma_N = \{y = \frac{1}{2}\}$, periodic boundary conditions on the left and right side, and the interface is implicitly defined by the zero level set of $\phi = \phi(x, y) =$

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

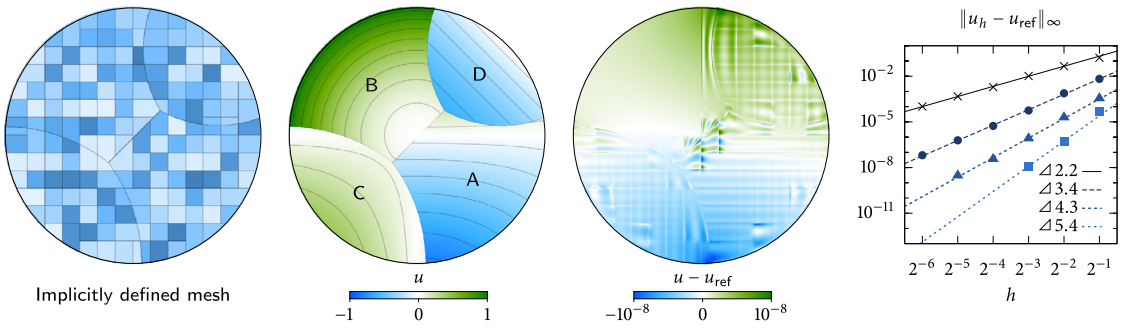

Fig. 6. Results of the grid convergence test problem (a) in §2.3.6. In the specific case of a 16×16 background Cartesian grid and $p = 4$ biquartic elements, the computed discrete solution u is a piecewise polynomial function defined on the depicted implicitly defined mesh; the middle-right figure plots the error in the discrete solution, which exhibits discontinuities in alignment with the element boundaries. The graph provides the results of the grid convergence analysis, showing the maximum-norm error in the discrete solution as a function of typical element size h and order p, comparing against the exact solution. Data points represent measured errors, whereas the lines of indicated slope illustrate the observed order of accuracy. $\times$ denotes $p = 1$, $\bullet$ denotes $p = 2$, $\blacktriangle$ denotes $p = 3$, $\blacksquare$ denotes $p = 4$.

$y - 0.1\cos(2\pi x)$. The exact solution is chosen to be $u = \cos 2\pi x \cos \frac{3}{2}\pi y$ or $u = \sin 2\pi x \sin \frac{3}{2}\pi y$, for phases one and two, respectively.

- In three dimensions, the domain is a cube, $\Omega = (-\frac{1}{2}, \frac{1}{2})^3$, Dirichlet boundary conditions are imposed on $\Gamma_D = \{y = -\frac{1}{2}\}$, Neumann boundary conditions on the opposite face, $\Gamma_N = \{y = \frac{1}{2}\}$, with periodic boundary conditions on all other sides. The interface is defined similarly to the two-dimensional problem, except its amplitude is modulated in the z-direction such that $\phi = y - 0.1(0.6 + 0.4\cos 2\pi z)\cos 2\pi x$. The exact solution is chosen to be $u = \cos 2\pi x \cos \frac{3}{2}\pi y \cos 2\pi z$, $u = \sin 2\pi x \sin \frac{3}{2}\pi y \sin 2\pi z$, for phases one and two, respectively.

In both 2D and 3D, phase one has ellipticity coefficient $\alpha = 1$, whereas phase two has $\alpha = 1000$; last, the interfacial jump data, boundary data, and right hand side source terms in (16) are defined corresponding to the chosen exact solution.

(c) *Heat operator, vector-valued, elliptic interface problem with phase-dependent coefficients.* In this last problem, we consider a simple variation of the vector-valued elliptic interface problem in (18), which arises in a backward Euler or Crank–Nicolson solve for a fluid with phase-dependent density and viscosity, whereby the solution $\mathbf{u} : \Omega \to \mathbb{R}^d$ to the following problem is sought:

$$\begin{cases} \beta_i \mathbf{u} - \nabla \cdot (\alpha_i(\nabla \mathbf{u} + \nabla \mathbf{u}^\intercal)) = \mathbf{f} & \text{in } \Omega_i \\ [\mathbf{u}] = \mathbf{g}_{ij} & \text{on } \Gamma_{ij} \\ [\alpha(\nabla \mathbf{u} + \nabla \mathbf{u}^\intercal)] \cdot \mathbf{n} = \mathbf{h}_{ij} & \text{on } \Gamma_{ij} \\ \mathbf{u} = \mathbf{g} & \text{on } \Gamma_D \\ (\alpha \nabla \mathbf{u}) \cdot \mathbf{n} = \mathbf{h} & \text{on } \Gamma_N. \end{cases} \tag{20}$$

Its discretisation is a simple modification of that described in the previous section §2.3.5, involving the addition of $M\mathcal{B}$ to the diagonal terms of the block form, where $\mathcal{B}$ is the diagonal matrix corresponding to the nodal interpolant of the phase-dependent coefficient $\beta > 0$, here assumed uniformly constant within each phase. The test problem domain, interface, and boundary conditions are identical to (b) above. In two dimensions, the exact solution is given by $\mathbf{u} = (u, v)$, where $u = \cos 2\pi x \cos \frac{3}{2}\pi y$, $v = \sin 2\pi x \sin \frac{3}{2}\pi y$ in phase one; $u = \sin 2\pi x \sin \frac{3}{2}\pi y$, $v = \cos 2\pi x \cos \frac{3}{2}\pi y$ in phase two. In three dimensions, the exact solution is given by $\mathbf{u} = (u, v, w)$, where $u = \cos 2\pi x \cos \frac{3}{2}\pi y \cos 2\pi z$, $v = \sin 2\pi x \sin \frac{3}{2}\pi y \sin 2\pi z$, $w = u$ in phase one; $u = \sin 2\pi x \sin \frac{3}{2}\pi y \sin 2\pi z$, $v = \cos 2\pi x \cos \frac{3}{2}\pi y \cos 2\pi z$, $w = u$ in phase two. Last, $\alpha_1 = 1$, $\alpha_2 = 16$, $\beta_1 = 1$, and $\beta_2 = 4$.

For each of these problems, a series of grid convergence tests is conducted, wherein the implicit mesh is defined by a background uniform Cartesian grid of size $2 \times 2\,(\times 2)$, $4 \times 4\,(\times 4)$, $8 \times 8\,(\times 8)$, etc., and errors in the computed discrete solution are measured in the maximum norm; denote by h as the cell size of the grid, and thus also the typical element size within the implicitly defined mesh. We consider $p = 1, 2, 3$, and 4, corresponding to bilinear, biquadratic, bicubic, and biquartic polynomials in two dimensions, and trilinear, triquadratic, tricubic, and triquartic polynomials in three dimensions. Corresponding to each problem:

(a) Fig. 6 illustrates the mesh in the case of a 16×16 background Cartesian grid, showing that implicitly defined meshes can also be applied to multiphase systems with triple junctions. Shown also is the computed discrete solution on the same mesh, using $p = 4$ biquartic elements, together with its pointwise error; the error has discontinuities in alignment with mesh boundaries, as one may expect in a dG method. The graph collects the results of the grid convergence study and shows that the maximum-norm error in the discrete solution is $\mathcal{O}(h^{p+1})$ for all p considered. (Here and in the

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

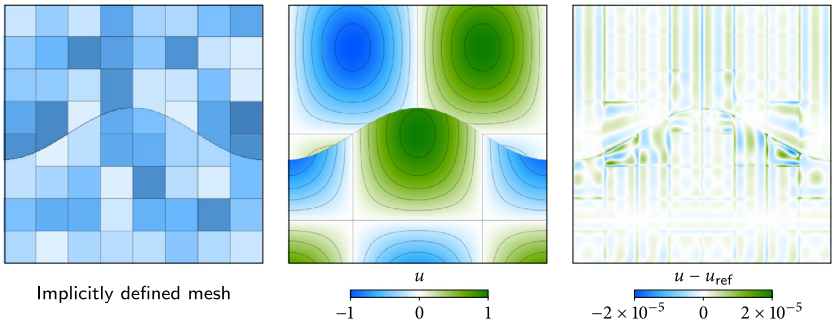

Fig. 7. Corresponding to the grid convergence test problem (b) in §2.3.6, in the specific case of a 8×8 background Cartesian grid and $p = 4$ biquartic elements, the computed discrete solution u is a piecewise polynomial function on the depicted implicitly defined mesh and is shown together with its pointwise error.

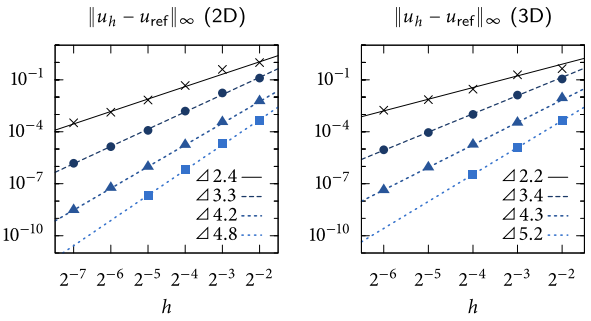

Fig. 8. Grid convergence results for test problem (b) in §2.3.6, showing the maximum-norm error in the discrete solution as a function of typical element size h and order p, comparing against the exact solution. Data points represent measured errors, whereas the lines of indicated slope illustrate the observed order of accuracy. × denotes $p = 1$, • denotes $p = 2$, ▲ denotes $p = 3$, ■ denotes $p = 4$.

following, i.e., in the graphs of Figs. 6, 8, and 9, some data points may appear "missing"; in these cases, corresponding to the smallest h and highest p, the error is saturated by double-precision arithmetic errors caused by the numerical conditioning of the linear system.)

(b) Fig. 7 and Fig. 8 show the results for test problem (b). The first figure illustrates the mesh and the discrete solution for $p = 4$ corresponding to a background 8×8 Cartesian grid, and the second figure graphs the maximum-norm error as a function of element size h and order p for the two-dimensional and three-dimensional test cases. Once again, the observed order of accuracy is $p + 1$, which is optimal for the degree p elements considered.

(c) Analogously, Fig. 9 contains the results for test problem (c). The experiments also show that the error in the maximum norm has order of accuracy approximately $p + 1$, in both two and three dimensions.

2.3.7. Projection operators

To conclude this section on LDG methods for elliptic operators on implicit meshes, we consider the discretisation of a certain class of projection operators used within incompressible fluid flow. These operators implement the Helmholtz–Hodge decomposition to decompose a given vector field $\mathbf{m} : \Omega \to \mathbb{R}^d$ as the sum of a divergence-free vector field and the gradient of a scalar field. To uniquely specify the decomposition, one must impose boundary conditions and interfacial jump conditions on each of these parts. We consider here the projection operator $\mathbb{P}$ such that $\mathbf{u} = \mathbb{P}(\mathbf{m})$ returns the divergence-free component of $\mathbf{m}$, where

$$\begin{cases} \mathbf{u} = \mathbf{m} - \nabla\varphi & \text{in } \Omega_i \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega_i \\ \mathbf{u} \cdot \mathbf{n} = u_n & \text{on } \partial\Omega \\ [\mathbf{u} \cdot \mathbf{n}] = 0 & \text{on } \Gamma_{ij}. \end{cases}$$

Here, $\varphi : \Omega \to \mathbb{R}$ is the aforementioned scalar field of the Helmholtz–Hodge decomposition, while $u_n : \partial\Omega \to \mathbb{R}$ is a given inflow/outflow boundary condition for the normal component of $\mathbf{u}$ on $\partial\Omega$. It follows that φ satisfies the elliptic interface problem

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

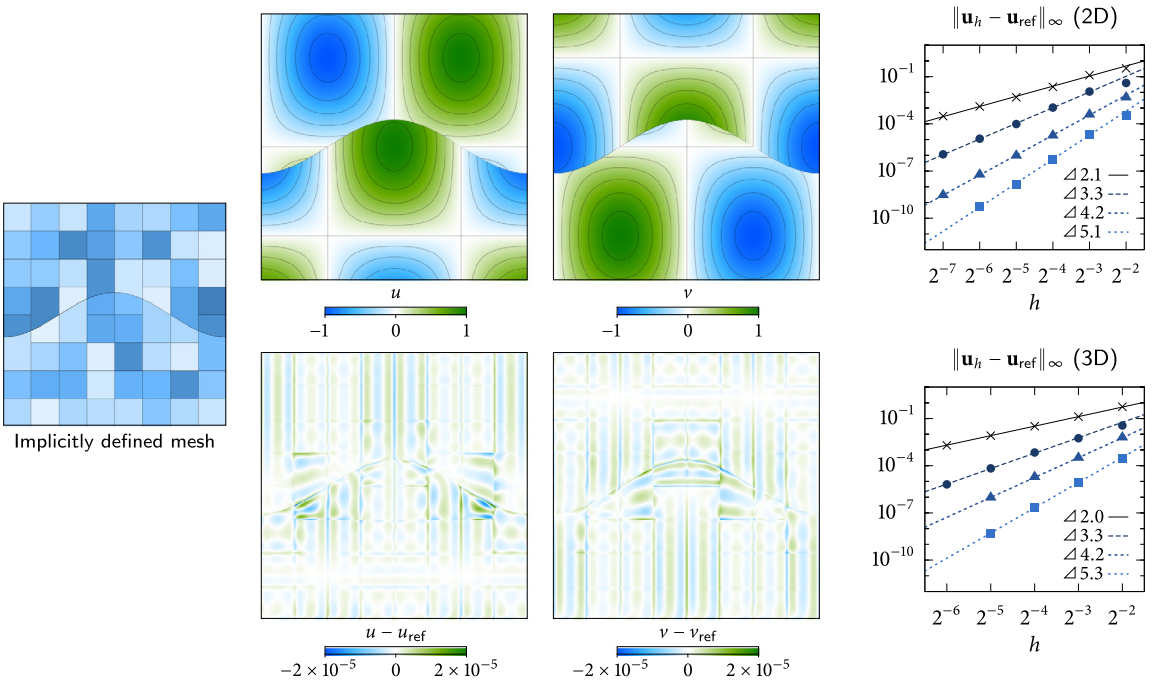

Fig. 9. Results of the grid convergence test problem (c) in §2.3.6. In the specific case of a 8×8 background Cartesian grid and $p = 4$ biquartic elements, the computed discrete solution $\mathbf{u} = (u, v)$ is a piecewise polynomial function defined on the depicted implicitly defined mesh; the figures plot the discrete solution together with its pointwise error. The graphs provide the results of the grid convergence analysis, showing the maximum-norm error in the discrete solution as a function of typical element size h and order p, comparing against the exact solution. Data points represent measured errors, whereas the lines of indicated slope illustrate the observed order of accuracy. × denotes $p = 1$, • denotes $p = 2$, ▲ denotes $p = 3$, ■ denotes $p = 4$.

$$\begin{cases} \Delta\varphi = \nabla\cdot\mathbf{m} & \text{in } \Omega_i \\ \partial_n\varphi = \mathbf{m}\cdot\mathbf{n} - u_n & \text{on } \partial\Omega \\ [\partial_n\varphi] = [\mathbf{m}\cdot\mathbf{n}] & \text{on } \Gamma_{ij}. \end{cases} \tag{21}$$

Note that this is not yet a well-posed problem for φ, as a condition on the interfacial jump $[\varphi]$ is missing: here we take for simplicity $[\varphi] = 0$ on Γ_{ij}. Appended with this extra condition, (21) uniquely determines φ up to a global constant; the constant, however, is irrelevant when computing $\mathbf{u} = \mathbf{m} - \nabla\varphi$.

In applications, it is typically the case that $\mathbf{m}$ is given as a piecewise polynomial vector field defined on the implicit mesh. It follows that one must calculate its discrete divergence in order to form the right hand side of the linear system corresponding to (21). For reasons shortly seen, it is particularly natural to compute $\nabla\cdot\mathbf{m}$ with the same discrete divergence operator as defined by the LDG method. Note though, since the interfacial jumps in $\mathbf{m}$ are a priori unknown, the jump data functional $J_h(h, h_{ij})$ required by (11) must be inferred from $\mathbf{m}$ itself. However, note also that precisely the same jump data is required to impose the interfacial jump conditions in (21) involving $[\mathbf{m}\cdot\mathbf{n}]$. In effect, these two sources of interfacial jump data cancel, and thus, by an appropriate modification to (15), we are left with the following linear system:

$$\Big[\Big(\sum_{1\le k\le d} G_k^{\mathsf{T}} M G_k\Big) + \tau_0 A_0 + \sum_{i>j} \tau_{ij} A_{ij}\Big]\varphi = \sum_{1\le k\le d} G_k^{\mathsf{T}} M m_k - M J(u_n), \tag{22}$$

where $m_i \in V_h$, $1 \le i \le d$, are the components of $\mathbf{m}$, and $J(u_n) \in V_h$ is the lifting of the u_n boundary data such that $\int_\Omega J(u_n) v = \int_{\partial\Omega} u_n v^-$ for all $v \in V_h$. Once (22) is inverted to determine φ, the discrete gradient of φ is then computed to find the projected velocity field $\mathbf{u} = \mathbf{m} - G\varphi$. The discrete projection operator $\mathbb{P}_h : V_h^d \to V_h^d$ is therefore given by

$$\mathbb{P}_h(\mathbf{m}) = \mathbf{m} - G\Big[\Big(\sum_{1\le k\le d} G_k^{\mathsf{T}} M G_k\Big) + \tau_0 A_0 + \sum_{i>j} \tau_{ij} A_{ij}\Big]^{-1}\Big(\sum_{1\le k\le d} G_k^{\mathsf{T}} M m_k - M J(u_n)\Big).$$

We note several aspects:

- Computing the discrete divergence of $\mathbf{u} = \mathbb{P}_h(\mathbf{m}) = \mathbf{m} - G\varphi$ using (11) and the boundary data for $\mathbf{u}$, one obtains

$$-\sum_{1\le k\le d} M^{-1} G_k^{\mathsf{T}} M u_k + J(u_n) = -\sum_{1\le k\le d} M^{-1} G_k^{\mathsf{T}} M(m_k - G_k\varphi) + J(u_n)$$

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

$$= M^{-1}\Big(-\sum_{1\le k\le d} G_k^{\mathsf{T}} M m_k + M J(u_n)\Big) + M^{-1}\sum_{1\le k\le d} G_k^{\mathsf{T}} M G_k \varphi$$
$$= -M^{-1}\Big[\tau_0 A_0 + \tau_{ij}\sum_{i>j}\tau_{ij} A_{ij}\Big]\varphi.$$

Therefore, if the penalty parameters are such that $\tau_0 = \tau_{ij} = 0$, or if φ is continuous across intraphase/interphase faces, the discrete divergence of $\mathbf{u}$ is zero. Hence, ignoring the influence of the penalty parameters, the projection operator is discretely idempotent, i.e., $\mathbb{P}_h\mathbb{P}_h = \mathbb{P}_h$. As discussed in §2.3.2, it is sometimes necessary to include penalty terms to ensure the linear system in (22) is well-posed, which in this context means that the kernel of the linear operator consists only of the constant functions. In this work, $\tau_0 = 0$ always sufficed, whereas interphase faces only needed a small amount of stabilisation, $\tau_{ij} = 0.1$. In practice, the computed φ is very nearly continuous across all faces, i.e., the magnitude of the discontinuity is a small fraction of $\mathcal{O}(h^{p+1})$. Hence, in practice, the discrete projection operator is essentially discretely idempotent.

- As remarked above, the discrete projection operator has the convenient property that neither $[\mathbf{m}\cdot\mathbf{n}]$ on Γ_{ij} nor $\mathbf{m}\cdot\mathbf{n}$ on Γ need to be computed or provided as source data. Experiments indicate that this property actually increases the stability and robustness of projection operators used within the interfacial gauge methods considered later in this work; in other words, even if jump conditions on $\mathbf{m}$ slowly "drift", the projected velocity always satisfies $[\mathbf{u}\cdot\mathbf{n}] = 0$ (up to discretisation errors), which aids in accurately evolving quantities by the fluid velocity $\mathbf{u}$.
- Another interpretation of the Helmholtz–Hodge decomposition, and in particular its discrete implementation, is to find a divergence-free vector field $\mathbf{u}$ that is closest to $\mathbf{m}$ in the L^2 norm. To see this, let $\boldsymbol{\delta} \in V_h^d$ be the smallest perturbation to $\mathbf{m}$ whose discrete divergence (with boundary data u_n) is zero, i.e., $M^{-1}\sum_k G_k^{\mathsf{T}} M(m_k + \delta_k) - J(u_n) = 0$. Using the method of Lagrange multipliers, since $\|\boldsymbol{\delta}\|^2 = \sum_k \delta_k^{\mathsf{T}} M \delta_k$, the problem amounts to minimising the functional $(\boldsymbol{\delta}, \lambda) \mapsto \frac{1}{2}\sum_k \delta_k^{\mathsf{T}} M\delta_k + (M^{-1}\sum_k G_k^{\mathsf{T}} M(m_k+\delta_k) - J(u_n))^{\mathsf{T}}\lambda$ where λ is a vector of length equal to the dimension of V_h. Straightforward manipulation shows that the minimum satisfies $\delta_k = -G_k M^{-1}\lambda$ and $\sum_k G_k^{\mathsf{T}} M G_k M^{-1}\lambda = \sum_k G_k^{\mathsf{T}} M m_k - M J(u_n)$. Note that, ignoring the penalty terms, this is equivalent to (22), once we identify $M^{-1}\lambda = \varphi$ and the correction as $\boldsymbol{\delta} = -GM^{-1}\lambda = -G\varphi$. Thus, putting aside the influence of the penalty parameters, one may also view the projection operator as an orthogonal projection onto the space of discretely divergence-free vector fields satisfying the boundary condition $\mathbf{u}\cdot\mathbf{n} = u_n$.

In general, the order of accuracy of the described discrete projection operator is as follows: assuming $\mathbf{m}$ is sufficiently smooth, φ attains full, optimal-order accuracy of $p+1$ in the maximum norm, whereas $\mathbf{u}$ attains one order less, i.e., a sub-optimal order of accuracy p in the maximum norm. Experiments indicate that the order reduction occurs wherever the mesh has non-uniform structure or at the boundary of the domain: for example, within or near curved elements, or at the boundaries of different levels of refinement within an adaptively refined quadtree/octree. Conversely, on a uniform Cartesian grid with periodic boundary conditions, the projection operator attains full order accuracy in both $\mathbf{u}$ and φ; this result is attributed to the special structure a Cartesian grid has in relation to the purely "one-sided" flux within the discrete divergence and gradient operators [55,56]. To demonstrate the order reduction, we consider here a simple grid convergence study: the test problem consists of finding the projection of $\mathbf{m} = (\sin 2x\cos 2y + ye^{xy}, -\cos 2x\sin 2y + xe^{xy})$ defined on the two-dimensional unit disc $\Omega = \{x^2 + y^2 < 1\}$ with boundary data $u_n = (\sin 2x\cos 2y, -\cos 2x\sin 2y)\cdot\mathbf{n}$; the exact solution satisfies $\mathbf{u} = \mathbb{P}(\mathbf{m}) = (\sin 2x\cos 2y, -\cos 2x\sin 2y)$ and $\varphi = e^{xy} + C$, where C is an arbitrary constant. Fig. 10 illustrates the discrete solution and its error in the specific case of an implicit mesh defined by a 16×16 background grid with $p = 4$ biquartic elements; note that the error in $\mathbf{u}$ has greatest magnitude near the boundaries of curved elements situated near $\partial\Omega$, and has a smaller error in the interior. (The arbitrary constant C in the gauge field is chosen such that the computed solution and exact solution have the same mean value $\int_\Omega \varphi$.) Fig. 11 collects the results of a grid convergence study for this problem, using grid sizes ranging from 8×8 to 128×128 and $p = 1, 2, 3$, and 4 elements, and shows the maximum-norm and L^2-norm error in the computed gauge field φ and $\mathbf{u}$. In both norms, φ attains optimal order accuracy, i.e., $p+1$. However, in the maximum norm, $\mathbf{u}$ has order of accuracy approximately p, and in the L^2 norm, order approximately $p + \frac{1}{2}$. (Note that the half-order increase in the L^2 norm is consistent with there being $\mathcal{O}(h^{-1})$ many boundary elements in which the solution is $\mathcal{O}(h^p)$ accurate, whereas the interior elements have order of accuracy approximately $\mathcal{O}(h^{p+1})$.) Similar conclusions hold for three-dimensional meshes as well as multiphase problems with interfaces.

2.4. Geometric multigrid

Before pursuing aspects of implicit mesh dG methods for time-dependent problems, we next consider efficient solution algorithms for the elliptic interface problems discussed in §2.3 by designing a set of geometric multigrid methods. When used as part of a standard V-cycle, these algorithms preserve the symmetric positive (semi)definite property of the discrete problems, allowing V-cycles to be used as a preconditioners for the conjugate gradient method. The combination of multigrid preconditioned conjugate gradient algorithm combines the advantages of both solvers: the multigrid preconditioner is efficient in the interior of the domain where the elliptic behaviour of the matrix dominates, while the conjugate gradient

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

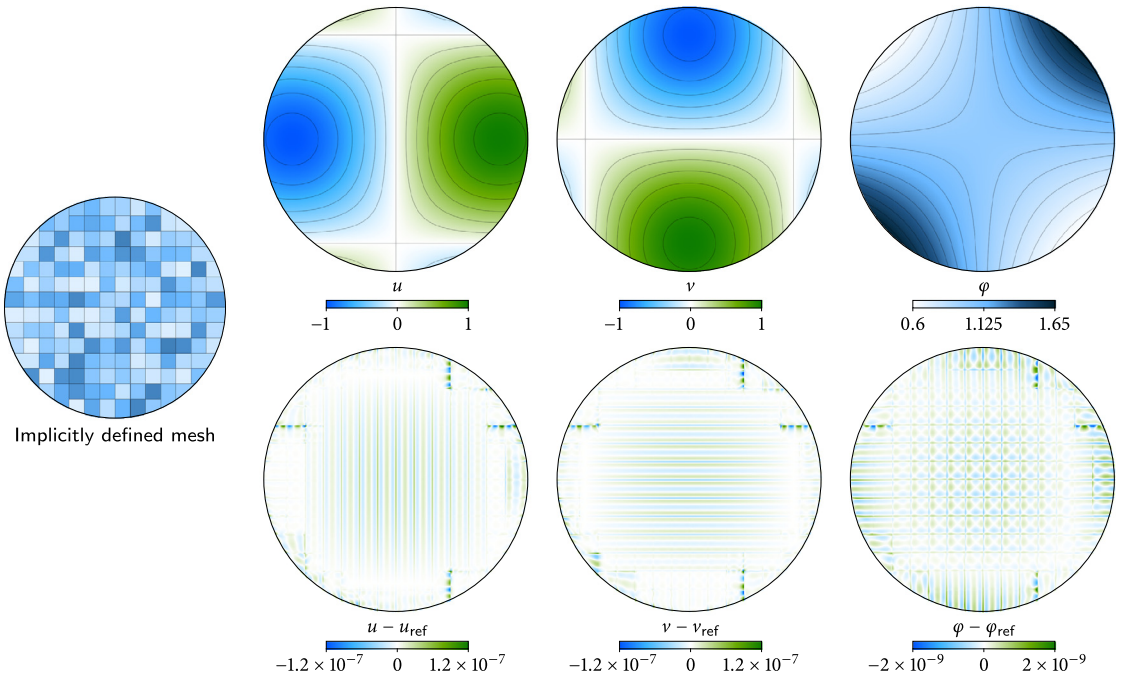

Fig. 10. Corresponding to the projection operator convergence study in §2.3.7, the discrete solution is shown in the specific case of an 16×16 background Cartesian grid and $p = 4$ biquartic elements. The top row illustrates the computed velocity field $\mathbf{u} = (u, v)$ and gauge field φ, which are each piecewise polynomial functions on the depicted implicitly defined mesh; the bottom row plots the error in the discrete solution, which exhibits discontinuities in alignment with the element boundaries.

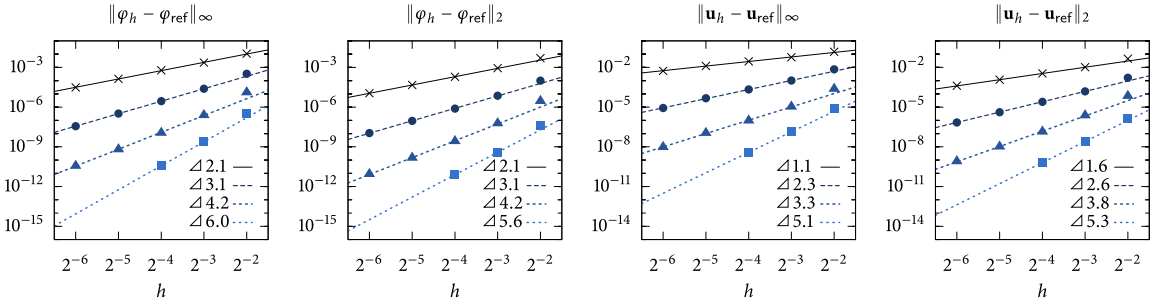

Fig. 11. Results of a grid convergence analysis for the projection operator showing the maximum-norm and L^2-norm error in the computed velocity field $\mathbf{u}$ and gauge field φ as a function of element size h and element order p, comparing against the exact (reference) solution. Data points represent measured errors, whereas the lines of indicated slope illustrate the observed order of accuracy. $\times$ denotes $p = 1$, $\bullet$ denotes $p = 2$, $\blacktriangle$ denotes $p = 3$, $\blacksquare$ denotes $p = 4$.

method effectively treats the remaining eigenmodes associated with imposing boundary conditions and interfacial jump conditions. Here we restrict attention to developing an h-multigrid method, i.e., one in which the mesh is coarsened at each level. p-multigrid methods are also possible, which hold the mesh fixed and instead coarsen the polynomial space by effectively reducing p each level. The latter methods are more efficient when high order dG methods are considered, and would likely be of benefit for the implicit mesh framework considered here, however they have not been considered as part of this work.

Briefly, the ingredients of a multigrid method are a mesh hierarchy, an interpolation operator to transfer approximate solutions from a coarse mesh onto a fine mesh, a restriction operator which helps formulate a coarse mesh correction problem by restricting the residual, and a smoother/relaxation method. We consider these first and then combine them into a V-cycle suitable for preconditioning the conjugate gradient method. Experimental results indicative of the overall convergence efficiency are then shown. It is assumed here that the reader is familiar with multigrid methods; see for example [57] for a review of their design and operation.

2.4.1. Mesh hierarchy

Recall that, in a geometric multigrid method applied to a hierarchy of meshes with progressively coarser resolution, the role of the coarse mesh problem is to (approximately) solve for a correction in the residual equation of a fine mesh problem. In particular, the coarse mesh problem solves an elliptic PDE problem identical to (or perhaps reminiscent of) the elliptic

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

PDE problem posed on the finest mesh. In the elliptic interface problems considered in this work, we have a choice as to whether to include the interfacial geometry on the coarse meshes. For example:

- In the simplest of elliptic interface problems,

$$-\Delta u = f \text{ in } \Omega \setminus \Gamma, \quad [u] = g \text{ on } \Gamma, \quad [\partial_n u] = h \text{ on } \Gamma,$$

  the interfacial jump data g and h is "lifted" into the interior in a way that is essentially equivalent to solving

$$-\Delta u = \tilde{f} \text{ in } \Omega, \quad [u] = 0 \text{ on } \Gamma, \quad [\partial_n u] = 0 \text{ on } \Gamma,$$

  where $\tilde{f}$ is equal to f plus source terms depending on g and h. In this case, because the coarse mesh problem assumes homogeneous interfacial jump conditions, it follows that the coarse mesh problem effectively solves a standard Poisson problem with uniform ellipticity coefficient. As a result, it is unnecessary for the coarse mesh to sharply represent the interface, i.e., the shape of coarse mesh elements can be oblivious to the location of the interface.
- In an elliptic interface problem with a jump in ellipticity coefficient, e.g.,

$$-\nabla \cdot (\alpha \nabla u) = f \text{ in } \Omega \setminus \Gamma, \quad [u] = g \text{ on } \Gamma, \quad [\alpha \partial_n u] = h \text{ on } \Gamma,$$

  where $[\alpha] \neq 0$, for optimal convergence properties it is ideal for the coarse mesh problem to have identical jumps in the ellipticity coefficient. In this case, the coarse mesh benefits from being interface conforming, i.e., elements are curved and sharply represent the interface.

Regarding the mesh hierarchy, in this work quadtrees/octrees are used to implicitly define the principal (finest) mesh; the tree naturally defines a hierarchy which coarsens the mesh by a factor of two in each dimension for each level. In particular, when the tree is used to implement adaptive mesh refinement (see §2.6.6), the cell size is not uniform throughout the domain. In this case, a variety of strategies could be used; here we consider two possibilities:

- *Rapid coarsening* – cells within the quadtree/octree are coarsened as rapidly as possible, i.e., the cells of the next-coarsest level consist of those nodes in the tree having all children cells members of the current level.
- *Height-based* – only the smallest cells of each level are coarsened to determine the next level of the hierarchy.

These two strategies are identical when the quadtree/octree is uniformly refined throughout the domain, but have different rates of removing degrees of freedom when the refinement is nonuniform.

The combination of the above four choices (two for whether or not implicitly defined coarse meshes are interface conforming, two for the coarsening strategy within the quadtree/octree itself) leads to four different possible mesh hierarchies, as illustrated by Fig. 12. A few remarks are in order:

- When the coarse meshes depend on the implicitly defined interface, a suitable definition of the level set function ϕ is required for the coarse grids of the quadtree/octree. Note however, whilst a fine mesh may have features in the interface with small length scales (e.g., small capillary waves), it is unnecessary for a very coarse mesh to represent the same length scales. Thus, one may also coarsen the level set function as one coarsens the mesh. It is unclear, however, what the optimal strategy may be in this regard. In this work, ϕ is defined on the coarse grid cells of the quadtree/octree as a piecewise tensor-product polynomial of degree p^d, the nodal values of which are interpolated from ϕ on the finest mesh.
- The coarse grid elliptic interface problem is constructed with the same LDG discretisations discussed in §2.3. On each mesh, this leads to a block-sparse symmetric positive (semi)definite matrix A.
- On a square/cubic domain, such that coarse meshes ignore interface geometry, one can coarsen the mesh all the way to one element covering the whole domain. When the domain has less simple geometry, it may be impractical to coarsen to one element; in this case, the hierarchy can be stopped short such that the coarsest mesh is a handful of elements, consequently relying more heavily on an efficient bottom solver within a multigrid V-cycle. On a related note, when the coarse meshes include interface geometry, it is possible that small phase cells will be encountered that have no appropriate neighbours with which to merge. One strategy to handle this issue is to halt the mesh hierarchy as soon as the situation arises. However, an alternate, more efficient strategy is to overrule the classification and force any such cells encountered to be considered "large", and continue the mesh hierarchy. Thus, while tiny cells are a significant problem in regards to ill-conditioning and time step constraints on the finest mesh, small cells in the coarsest of meshes in the multigrid hierarchy do not appear to be a problem.

There are a wide variety of additional factors that play a role in the efficiency of a geometric multigrid method. Here our primary focus is on a simple implementation, and as such we forgo additional considerations.

2.4.2. Interpolation operator

The interpolation operator transfers a piecewise polynomial function $u_c \in V_{2h}(\Omega_c)$ (defined on a coarse mesh) to a piecewise polynomial function $u_f \in V_h(\Omega_f)$ (defined on a fine mesh) via some kind of interpolation. Note that u_c may

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

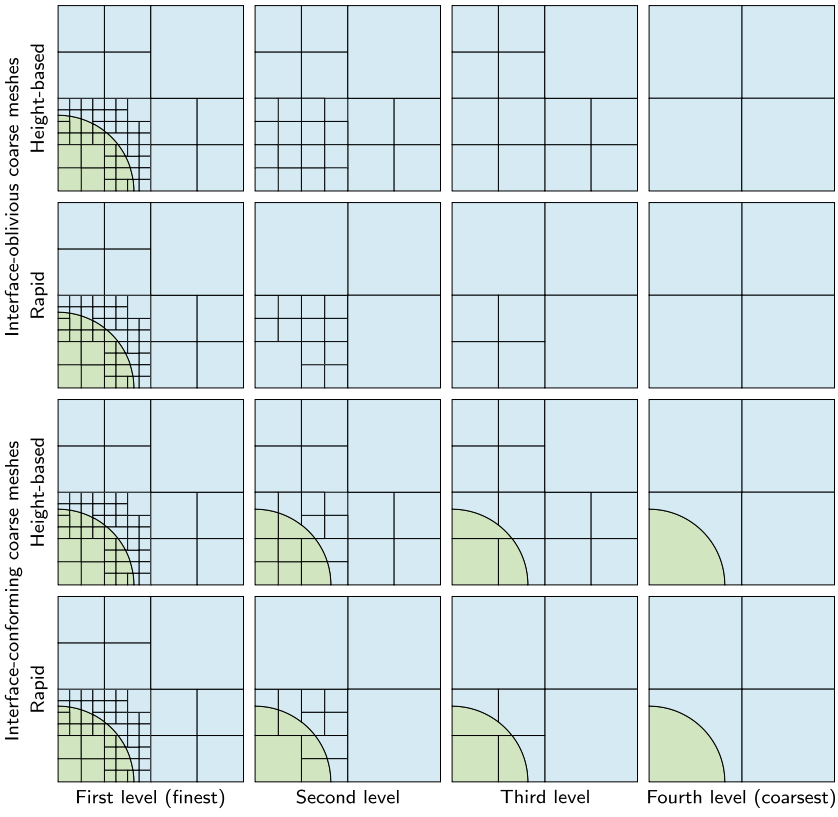

Fig. 12. Mesh hierarchy possibilities for geometric multigrid, with each row illustrating a different option. Top two rows: coarser levels in the hierarchy are oblivious to the location of the interface. Bottom two rows: coarser level meshes are implicitly defined by a coarse representation of the interface. In the height-based strategy, only the smallest cells of the current quadtree level are merged to define the next level; in the rapid coarsening strategy, children cells of a parent node in the quadtree are merged provided they are all leaf nodes.

be discontinuous across element boundaries, and so one must define an operator appropriately treating this ambiguity; a natural approach is to adopt the geometric structure of the quadtree/octree, as follows. To define $u_f|_{E_f}$ for a fine mesh element E_f, note that E_f has its degrees of freedom associated to a "parent cell" c in the tree (as discussed in §2.2.5). The parent $P(c)$ of c in the quadtree/octree hierarchy on the next coarsest level is then found. $P(c)$, whether it is small, large, or entire in the nomenclature of the implicit mesh defined on the coarse grid, must necessarily be associated with an element E_c on the coarse mesh;[7] it is this element which the fine mesh's element interpolates from, i.e., $u_f|_{E_f}$ is defined to be the same polynomial as $u_c|_{E_c}$. Thus, with the aid of the tree structure of the quadtree/octree background grid, this defines an interpolation operator $I_c^f : V_{2h}(\Omega_c) \to V_h(\Omega_f)$. The operator is linear and has the property that it preserves constant functions, i.e., $u_c \equiv 1$ is mapped to $u_f \equiv 1$. This property ensures that, throughout a V-cycle, the coarse mesh discrete problems preserves the compatibility condition required in semidefinite elliptic interface problems (having only Neumann boundary conditions), as discussed later.

2.4.3. Restriction operator

The restriction operator $R_f^c : V_h(\Omega_f) \to V_{2h}(\Omega_c)$ is defined to be the adjoint of the interpolation operator, i.e., such that

$$(R_f^c u_f, u_c)_{\Omega_c} = (u_f, I_c^f u_c)_{\Omega_f} \tag{23}$$

holds for every $u_f \in V_h(\Omega_f)$ and every $u_c \in V_{2h}(\Omega_c)$. Abusing notation and letting I_c^f, R_f^c, u_c, and u_f also denote the matrices and vectors relative to the nodal basis of $V_h(\Omega_f)$ and $V_{2h}(\Omega_c)$, (23) implies that

$$(R_f^c u_f)^\intercal M_c u_c = u_f^\intercal M_f I_c^f u_c,$$

where M_c and M_f are the block-diagonal mass matrices of the coarse and fine mesh, respectively. Thus, in the nodal basis,

[7] When the coarse levels of the mesh hierarchy are multiphase, i.e., conform to the interface, the coarse mesh element attached to $P(c)$ having the same phase as c is used.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

Algorithm 1 One application of the relaxation/smoothing operator $\mathcal{S}_{\leftrightarrows}(x, b)$, given a matrix A with blocks A_{ij} and damping parameter ω.

for each processor p **do**
  Let $\mathcal{I} = (i_j)_{j=1}^{n_p}$ denote the user-chosen ordering of the n_p elements owned by processor p.
  If smoothing operator is called with reverse order, $\mathcal{S}_{\leftarrow}$, reverse the order of $\mathcal{I}$.
  Make a local copy of x.
  for $i \in \mathcal{I}$ (in order) **do**
    Solve for α such that $A_{ii}\alpha = b_i - \sum_{j \neq i} A_{ij} x_j$.
    Replace $x_i \leftarrow (1-\omega)x_i + \omega\alpha$.
return x as computed by each processor.

$$R_f^c = M_c^{-1} (I_c^f)^{\intercal} M_f. \tag{24}$$

An interpretation of the restriction operator is that it averages elemental polynomials on the fine mesh to determine a coarsened piecewise-polynomial representation of u_f on the coarse mesh. This averaging automatically handles cases where the fine mesh and coarse mesh have different structure (for example, near the interface or domain boundary where different cell merging decisions may have taken place, or near boundaries of different resolution in a mesh with adaptive mesh refinement). Moreover, the averaging is performed in a way that locally preserves the "mass" of u_f: indeed, since I_c^f preserves constant functions, we have that $(R_f^c u_f, 1)_{\Omega_c} = (u_f, 1)_{\Omega_f}$ for all $u_f \in V_h(\Omega_f)$.

2.4.4. Relaxation

Recall that the LDG discretisation of the elliptic interface problems in §2.3 results in a symmetric positive (semi)definite block-sparse matrix A, where each nonzero (i, j) block of A represents a dependence between element i and j. For the relaxation/smoothing method of multigrid, in this work a block Gauss–Seidel smoother is adopted (in fact, successive over-relaxation), considering two possible orderings for Gauss–Seidel:

- *Red-black ordering.* On a simple uniform Cartesian grid mesh, the LDG discretisation (with one-sided directional fluxes as used in §2.3) results in a linear operator with a block red-black property. Thus, one may expect a red-black Gauss–Seidel method to offer excellent smoothing properties. However, since the implicit meshes used in this work have more complicated connectivity near the interface or curved boundary, the corresponding discrete Laplacian does not in general have the red-black property. Nevertheless, one may define an ordering based on the colour of the cells in the quadtree/octree reminiscent of a red-black ordering.
- *Z-ordering.* A natural ordering of the elements in an implicit mesh comes from the depth-first traversal of their corresponding parent cells in the quadtree/octree. This results in an ordering similar to the Z-order curve.

We consider two more aspects before defining the relaxation algorithm. First, in order to use a block Gauss–Seidel method, the diagonal blocks of the discrete Laplacian must be inverted. One possibility, adopted here, is to precompute a Cholesky decomposition of each diagonal block. (The diagonal blocks of a symmetric positive (semi)definite matrix must themselves be symmetric positive (semi)definite.) According to experiment, precomputing the Cholesky factorisation of the block-diagonal matrix has negligible cost compared to the overall multigrid algorithm.

Second, with a view to a simple parallelisation of the multigrid algorithms, the relaxation is mildly damped, thereby replacing the Gauss–Seidel scheme with a form of successive over relaxation (SOR), as follows. Except for special cases like red-black ordering on a highly structured block domain decomposition (which as previously remarked does not apply to implicit meshes), Gauss–Seidel relaxation does not lend itself to a straightforward parallelisation [58]. Instead of adopting subtle synchronisation strategies (see, e.g., [59]) consider applying Gauss–Seidel locally to the set of elements on each processor, synchronising neighbouring "ghost" elements at the end of each pass (referred to as processor block Gauss–Seidel by Adams et al. [58]). Note, however, in the limit of one element per processor, this strategy collapses to a block-Jacobi smoother. As an iterative solver, block-Jacobi is not guaranteed to converge; indeed, it often fails to converge for the discrete Laplacians considered in this work. On the other hand, damped block-Jacobi converges, provided the damping factor is suitably chosen. Thus, to ensure convergence, even in the case that there are multiple elements per processor, we here dampen the Gauss–Seidel method according to an SOR scheme.

The resulting relaxation operator is defined in Algorithm 1. A few brief remarks are in order:

- $\mathcal{S}_{\rightarrow}$ denotes SOR relaxation in which the elements (for each processor p) are visited in a fixed, user-chosen order; $\mathcal{S}_{\leftarrow}$ denotes SOR relaxation with the reverse ordering. As discussed shortly, the ability to reverse the order is needed when designing a symmetric multigrid V-cycle.
- On a uniform Cartesian grid, experiments indicate the difference in efficiency between red-black ordering and Z-orderings is not significant. (However, this was not extensively tested, and may involve a variety of factors, such as damping, cache effects, number of processors, etc.)
- On implicitly defined meshes with curved boundaries and interfaces, the Z-ordering outperformed the red-black ordering in all test cases.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

Algorithm 2 Multigrid V-cycle $V(x_f, b_f)$ on fine mesh Ω_f, with ν pre-smoothing and post-smoothing steps (ν_c in the case of the coarsest mesh).

if Ω_f is the coarsest mesh **then**
  repeat ν_c times, $x_f = S_{\rightarrow}(x_f, b_f)$.
  repeat ν_c times, $x_f = S_{\leftarrow}(x_f, b_f)$.
else
  repeat ν times, $x_f = S_{\rightarrow}(x_f, b_f)$.
  Restrict and solve correction equation on coarse mesh to compute $x_c = V\big(0, (I_c^f)^{\intercal}(A_f x_f - b_f)\big)$.
  Interpolate and update $x_f = x_f - I_c^f x_c$.
  repeat ν times, $x_f = S_{\leftarrow}(x_f, b_f)$.
return x_f.

- In the present work, the SOR parameter is fixed to $\omega = 0.8$. Without parallelisation (i.e., only one processor), $\omega = 1$ is optimal; as more processors are added, it is necessary to reduce ω to ensure convergence. A value of 0.8 was found to be approximately the largest possible while reliably obtaining convergence independent of the number of processors. At this value, convergence efficiency does not appear to be markedly impacted (when coupled to multigrid preconditioned conjugate gradient methods), whereas values less than 0.7 tend to slow convergence.

2.4.5. Multigrid preconditioned conjugate gradient

Recall that a geometric multigrid method utilises a combination of relaxation/smoothing together with interpolated approximate solutions of coarse-grained problems. In the present case, the coarse-grained problem solves for the correction in a residual equation for the same elliptic problem except on a coarser mesh. In particular, it is important to note that the LDG methods developed in §2.3 operate in the "mass basis", i.e., instead of solving $\Delta_h u = f$, where Δ_h is the discrete Laplacian, one instead solves $M\Delta_h u = Mf$, as the latter system is symmetric positive (semi)definite. Thus, to appropriately define the coarse mesh problem, one may: (i) calculate the residual of the fine mesh linear system $A_f x_f = b_f$, (ii) multiply the residual by M^{-1} to correctly determine the residual as a piecewise polynomial function, (iii) restrict the residual to the coarse mesh, and then (iv) multiply this residual by M of the coarse mesh. Thus, the coarse mesh problem consists of (approximately) solving for x_c such that

$$A_c x_c = M_c\big(R_f^c\big[M_f^{-1}(A_f x_f - b_f)\big]\big).$$

In fact, according to the derived restriction operator (24), this conveniently simplifies to

$$A_c x_c = (I_c^f)^{\intercal}(A_f x_f - b_f),$$

and so it is unnecessary to multiply by mass matrices in the implementation of the multigrid method; instead, one can simply apply the transpose of the interpolation matrix. With this consideration in mind, the design of a multigrid V-cycle is relatively straightforward and is outlined in Algorithm 2. We note several important aspects:

- Called from the finest mesh of the hierarchy, with initial guess zero, the V-cycle is linear in its argument b. To use it as a preconditioner in a conjugate gradient method, it is necessary for the associated linear operator to be symmetric positive semidefinite. To ensure symmetry in particular, it is necessary to reverse the order of elements visited in the SOR/Gauss–Seidel algorithm as follows: on travelling down the V-cycle (pre-smoothing), a fixed order is used, i.e., $S_{\rightarrow}$; on travelling up the V-cycle (post-smoothing), the reversed order is used, i.e., $S_{\leftarrow}$. This results in a symmetric operator for an arbitrary number of parallel processors and arbitrary domain decomposition. It is outside the scope of this work to prove that the resulting V-cycle is indeed symmetric positive (semi)definite, though in practice it always has this property.
- The V-cycle consists of ν pre-smoothing and ν post-smoothing steps at each level of the mesh hierarchy. On the coarsest mesh of the hierarchy, a variety of options exist for solving $A_f x_f = b_f$, including applying a direct method, a conjugate gradient algorithm, or even setting $x_f = 0$. In this work, $\nu_c = 4$ smoothing steps are applied, having nearly identical convergence rates compared to using an exact solver on the coarsest mesh but with improved efficiency.
- Experiments indicate that $\nu = 2$ or $\nu = 3$ provides the best efficiency when coupled to a preconditioned conjugate gradient algorithm. (Note that, although larger ν typically leads to accelerated convergence rates, doing so comes at the expense of more smoothing steps and hence more computational effort.) It is noted that the design and implementation of the geometric multigrid algorithm is relatively simple; more efficient algorithms may be possible with the aid of more sophisticated relaxation methods, W-cycles, F-cycles, attention to memory cache effects, etc., but these aspects are not pursued further here. In fact, for $p \geq 3$ it is expected that p-multigrid would provide the largest reward in improving computational efficiency, as opposed to optimising the described h-multigrid method.
- Recall that, in the case of Neumann boundary conditions, more precisely when Γ_D is empty, the elliptic interface problems of §2.3 are positive semidefinite, with kernel equal to the space of constant functions. The convergence properties of both multigrid and conjugate gradient are unaffected by this zero eigenmode, provided the right hand side satisfies the compatibility condition $(f, 1) = 0$. Assuming f has mean zero on the finest mesh, it follows that the residual

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

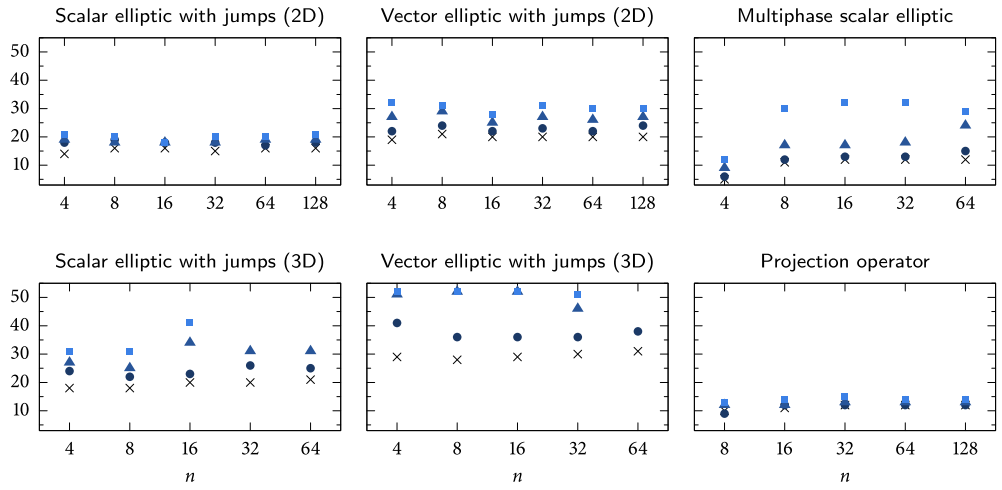

Fig. 13. Iteration counts for the multigrid-preconditioned conjugate gradient algorithm applied to a variety of elliptic PDE problems, plotting the number of iterations required to reduce the norm of the residual by a factor of 10^{10}; in all cases, the initial iterate (i.e., initial guess to the linear system) is the zero vector. n denotes the size of the background Cartesian grid used, i.e., $n \times n$ in 2D or $n \times n \times n$ in 3D. The left column corresponds to problem (b) of §2.3.6, the middle column corresponds to problem (c), top-right corresponds to problem (a), and bottom-right to the convergence test in §2.3.7. × denotes $p = 1$, • denotes $p = 2$, ▲ denotes $p = 3$, ■ denotes $p = 4$. (Data points which appear "missing" correspond to cases in which the conditioning of the linear system exceeds the operating limits of the conjugate gradient algorithm in double-precision arithmetic for the 10^{10} threshold, i.e., the order of magnitude of the condition number approximately exceeds 10^6.)

(for any approximate solution) on the finest mesh also has mean zero. Therefore, since $(R_f^c u, 1)_{\Omega_c} = (u, 1)_{\Omega_f}$ for all $u \in V_h(\Omega_f)$, it follows that the coarse mesh problem automatically satisfies the compatibility condition. Applied recursively, the compatibility condition is thus automatically satisfied throughout the V-cycle, which ensures that the V-cycle by itself converges and that it can also be used as a preconditioner for conjugate gradient. It is sometimes necessary, however, to enforce the compatibility condition discretely due to the accumulation of round-off error—in practice, the right hand side f is projected onto the space of mean-zero functions, for the finest mesh of the hierarchy as well as the coarsest mesh, but not for any of the levels in between.

It is straightforward to incorporate the V-cycle as a preconditioner in a conjugate gradient method: the resulting algorithm is used for all of the elliptic interface problems presented in this work. Generally speaking, the number of steps required to reduce the norm of the residual by a constant factor is independent of the number of elements or processor count, and thus the solution algorithm has optimal computational complexity. To illustrate, Fig. 13 plots the recorded number of iterations—using a threshold of 10^{10} reduction in the residual norm—for each of the grid convergence studies considered in §2.3.6 and §2.3.7. (For problem (a), having uniform ellipticity coefficient, the mesh hierarchy used coarse meshes oblivious to the interface location; for problems (b) and (c), which have jumps in the ellipticity coefficient, coarse meshes are interface-conforming.) The results collected in Fig. 13 reflect a variety of trends consistently observed throughout this work: elliptic interface problems with larger p require more iterations than those with smaller p; the number of iterations is approximately independent of the element size h; problems with uniform ellipticity coefficient generally require less iterations than those with jumps in the ellipticity coefficient (note, though, test problem (b) had a large factor of 1000 change in the ellipticity coefficient, yet the multigrid algorithm retains reasonable efficiency); and 3D problems require more iterations than 2D problems. Recall that the presented algorithm is based solely on h-multigrid, i.e., only the mesh is coarsened each level. It is quite likely that further improvements could be gained by using p-multigrid strategies in addition to h-multigrid, however this is not pursued further in this work.

2.5. Advection

In this section we briefly consider the discretisation of advective terms in conservative form. In the present work, this is required in two places: (i) in the advection of the level set function ϕ by fluid velocity, $\nabla \cdot (\mathbf{u}\phi)$, and (ii) in the advection term of the Navier–Stokes equations, $\nabla \cdot (\mathbf{u}u_i)$, $1 \le i \le d$. In both cases, the velocity field $\mathbf{u} = (u_1, \ldots, u_d)$ is assumed to have zero divergence.

Let $A(\mathbf{u}, \psi)$ denote the discretisation of $\nabla \cdot (\mathbf{u}\psi)$ for $\mathbf{u} \in V_h^d$ and $\psi \in V_h$. $A \in V_h$ is defined weakly such that

$$\int_E A(\mathbf{u}, \psi)\, v = -\int_E \psi\, \mathbf{u} \cdot \nabla v + \int_{\partial E} v\, (\mathbf{u}\psi)^\star \cdot \mathbf{n} \tag{25}$$

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

holds for every element $E \in \mathcal{E}$ and every test function $v \in V_h$. Here, $(\mathbf{u}\psi)^\star$ denotes a numerical flux which can take on a variety of forms: here, we employ an upwind flux and a (local) Lax–Friedrichs flux. In the following, for simplicity of presentation, these fluxes are defined to be vector-valued; however, only the normal component of the flux is used in (25).

For the upwinding flux, consider three cases: internal mesh faces, faces with inflow boundary conditions, and faces with outflow boundary conditions. On internal faces (i.e., either an intraphase or interphase face), define (see, e.g., [60])

$$(\mathbf{u}\psi)^\star_{\text{upwind}} := \begin{cases} u_L \psi_L \mathbf{n} & \text{if } u_L > 0 \text{ and } u_L + u_R > 0, \\ u_R \psi_R \mathbf{n} & \text{if } u_R < 0 \text{ and } u_L + u_R < 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $u_L := \mathbf{u}^- \cdot \mathbf{n}$, $u_R := \mathbf{u}^+ \cdot \mathbf{n}$, $\psi_L = \psi^-$, $\psi_R = \psi^+$, and $\mathbf{n}$ is the unit normal of the face. On outflow boundary faces, we define $(\mathbf{u}\psi)^\star := \mathbf{u}^-\psi^-$, and for inflow boundary faces, $(\mathbf{u}\psi)^\star := \mathbf{u}_\partial \psi_\partial$, where $\mathbf{u}_\partial$ and ψ_∂ are given boundary conditions for $\mathbf{u}$ and ψ, respectively (though, only the normal component of $\mathbf{u}_\partial$ needs definition). For the local Lax–Friedrichs flux, on internal faces (i.e., either intraphase or interphase) define

$$(\mathbf{u}\psi)^\star_{\text{LLF}} := \tfrac{1}{2}(\psi_L \mathbf{u}_L + \psi_R \mathbf{u}_R) + C \max(|\mathbf{u}_L \cdot \mathbf{n}|, |\mathbf{u}_R \cdot \mathbf{n}|)\,(\psi_L - \psi_R)\mathbf{n},$$

whereas for inflow and outflow boundary faces, the same values as those employed by the upwind flux are adopted. Here, $C \geq 1$ is a constant that factors into the evolutionary stability of a scheme which solves, e.g., $\psi_t + \nabla \cdot (\mathbf{u}\psi) = 0$, and in this work is set to $C = 1.5$. Two remarks are in order:

- The Lax–Friedrichs flux is often not the preferred choice when solving a hyperbolic conservation law, as it has a comparatively large amount of numerical dissipation. On the other hand, the Lax–Friedrichs flux often introduces less discontinuity than other choices of flux, and consequently may be more apt when the advected quantity is expected to be smooth. In fact, for the purposes of advecting the level set function, it was found in this work that the Lax–Friedrichs flux performed better than alternatives.
- Note that the governing integrals on the right hand side of (25) involve the product of three polynomials in V_h. To avoid aliasing errors (a form of variational crime, see, e.g., [28]) it is necessary to compute these integrals with quadrature schemes of sufficient accuracy. For rectangular elements and rectangular faces, a tensor-product Gaussian quadrature scheme with $\lceil \frac{3p+1}{2} \rceil$ points per dimension is used. For all other elements and faces, the precomputed quadrature schemes determined as part of the construction of the implicit mesh (see §2.2.2) are re-used. Thus, it is implicitly assumed that the corresponding quadrature schemes are sufficiently accurate; this aspect can be controlled with the user-chosen parameter q in the algorithms of [29].

2.6. Evolving meshes

We consider now a variety of topics relating to evolving interfaces, and thus evolving meshes. It is conceptually straightforward to incorporate an evolving interface into the implicit mesh dG framework: at every time step, the mesh is implicitly defined by the background quadtree/octree, together with the evolving level set function. Elements near the interface therefore change shape, and so their mass matrices and face integration rules are recomputed every time step. Since the interface does not move significantly between one time step and the next, the nodal basis of V_h corresponding to the mesh at one time step and the nodal basis of V_h for the mesh at the next time step is, for the most part, identical.[8] Consequently, the state (fluid velocity, pressure, etc.) can be injected from one mesh into the next, in essentially the same way as an Eulerian method; compare this to, e.g., Lagrangian or ALE methods, wherein the degrees of freedom move over time. Note, however, evolving element geometry implies changing discrete differential operators. In designing time stepping methods for the accurate computation of interface dynamics, one must therefore use the correct discrete operator applied at the appropriate point in time. We return to this topic later when specific applications are considered. In this section, our focus is on establishing the essential tools needed for an evolving interface problem, including: (i) choosing how to advect the level set function, (ii) considerations relating to computing high-order accurate distance functions, (iii) representing the level set function on the quadtree/octree to implicitly define the mesh, (iv) transferring state between one mesh and the next, and, last (v) a brief discussion regarding adaptive mesh refinement. In much of the following, for simplicity of presentation, we consider multiphase problems with at most two phases, and assume that the interface is implicitly defined by a single level set function ϕ.

2.6.1. Interface evolution

In implicit interface methods, such as the level set method [8,61,62] and the Voronoi implicit interface method [49,63], the evolution of the interface is determined by advecting a level set function ϕ using an appropriately defined extension velocity field [61,64]. The chosen velocity field must necessarily impose the correct physical speed of the interface, on the interface; elsewhere, it must be appropriately extended. There are two natural choices for the construction:

[8] A more precise account of changing mesh topology is given later in §2.6.4.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

- *Extension by adopting the physical velocity field* – In this case, ϕ is advected with the same velocity field as that given by the physics (e.g., the Navier–Stokes equations). It is important to note that—since the physical velocity field is usually not smooth across the interface, i.e., its higher-order derivatives may exhibit jumps—ϕ will become non-smooth across the interface. For example, in two-phase fluid flow driven by surface tension, in which the two phases have different fluid viscosities, the velocity field is continuous across Γ, but $\nabla \mathbf{u}$ generally exhibits jumps across Γ; consequently, ϕ will develop jumps in its gradient at the interface, the magnitude of which grows in time.
- *Extension by closest point* – in this case, one defines an extension velocity field $\mathbf{v}$ by $\mathbf{v}(x,t) := \mathbf{u}(\mathrm{cp}(x,t),t)$ and solves the advection equation $\phi_t + \mathbf{v} \cdot \nabla \phi = 0$, where $\mathrm{cp}(x,t)$ denotes the closest point to x on the interface at time t. Advection by this extension velocity has the property that if ϕ is a distance function at time $t = 0$, then it remains a distance function for all time [64]. Consequently, the evolving level set function exhibits less shearing than in the first option: in many cases, the technique offers improved numerical accuracy, such as in the numerical conservation of mass. Although one must construct the velocity field every time step, there are fast methods for doing so [64,65] incurring little computational expense. However, this method also has a strong reliance on being able to accurately interpolate the physical velocity at the interface, which means one must also take into account the non-smoothness of the physical velocity field.

In choosing one of these techniques for the implicit mesh dG framework, we recall an important aspect of the framework: state quantities, such as fluid velocity, are representable to high-order accuracy (assuming they are sufficiently smooth on either side of the interface), because the elements of the mesh are interface conforming, i.e., the interface is sharply represented. Thus, of the above two options, it is particularly natural to choose the first, wherein the level set function is represented as a piecewise polynomial function on the implicitly defined mesh. There are at least two reasons for doing so. First, it is natural to solve advection equations in a finite element method when the quantities involved are defined on the same mesh. Second, although the physical velocity is computed to high-order accuracy, its pointwise error is usually greatest on the curved interphase faces (see, e.g., Fig. 7 of part two [3]). It follows that the method of closest point extension would propagate these inaccuracies throughout the domain. On the other hand, advecting a piecewise polynomial level set function with the piecewise polynomial physical velocity results in an advection that has a weaker dependence on the accuracy at interphase faces. Again, it is emphasised that the resulting advected level set function will generally exhibit jumps in its derivatives across the interface, however it can nevertheless be computed with high-order accuracy.

2.6.2. High-order accurate reinitialisation

In theory, one may evolve the level set function using the physical velocity for an arbitrary time interval. However, it is often the case that physical currents and boundary effects cause the level set function to undergo a large amount of shearing, leading to large gradients and, eventually, decreased numerical accuracy. Thus, in practice, it is often advisable to periodically *reinitialise* the level set function by replacing it with a new function, usually the distance function to the interface. Such a reinitialisation procedure is nowadays commonplace in level set methods [61,62].

Here, the high-order accurate algorithm of [65] is adopted to reinitialise the level set function. Briefly, the algorithm works by first sampling the zero level set of ϕ to create a cloud of points on the interface having approximately uniform density. These "seed points" are approximately a distance Ch from one another, where $0.1 \lessapprox C \lessapprox 0.5$ and h is the local element size. Then, to compute the closest point on the interface to a given arbitrary point $x_0 \in \mathbb{R}^d$, one (i) finds the closest seed point x to x_0 using a k–d tree, and then (ii) improves x by inputting it into Newton's method to compute the true closest point on the interface of the polynomial that locally defines ϕ. Since the seed point is a good approximation to the true closest point, Newton's method typically converges to machine precision in as few as 2–3 iterations. The open source algorithms of [65], which include a k–d tree optimised for codimension-one point clouds, have been implemented in C++ and are freely available.

In this application, the level set function being reinitialised is an element of V_h, i.e., piecewise polynomial. To seed the point cloud, one can conveniently make use of the already-computed quadrature schemes for interphase faces by reusing the associated quadrature nodes to define the seed points. Within Newton's method, which begins with the closest seed point x to a given input x_0, the polynomial that implicitly defines the interface must be specified: since we know that x belongs to an interphase face, we define the required polynomial as the average of the two elemental polynomials defining ϕ on either side of the face. With these implementation details, the algorithm in [65] evaluates a high-order approximation to the closest point function, $\mathrm{cp} : \mathbb{R}^d \to \mathbb{R}^d$. In turn, this defines the signed distance function, $d_\phi : \mathbb{R}^d \to \mathbb{R}$, defined by $d_\phi(x) = \pm \|x - \mathrm{cp}(x)\|$, where the sign is chosen appropriately according to the phase of x. Due to the optimised k–d tree and the efficient Newton's method, experiments show that the computational cost of the algorithm is in practical settings insignificant. Moreover, since the computed closest point is (to machine precision) the exact closest point corresponding to the given polynomial, it follows that the order of accuracy of d_ϕ is at least $p + 1$ [65]; for example, one can demonstrate sixth order accurate reinitialisation using degree five polynomials.

To reinitialise ϕ, we replace ϕ with the L^2 projection of d_ϕ, i.e., for each element $E \in \mathcal{E}$, we define

$$\phi|_E = \underset{u \in \mathcal{Q}_p(E)}{\arg\min} \int_E (u - d_\phi)^2,$$

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

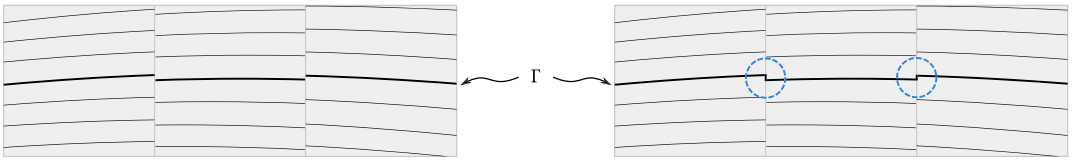

Fig. 14. A simplified and exaggerated depiction of the discontinuities introduced in a level set function as a result of advection via dG methods. Three square elements are shown, together with the contours of a fictitious level set function on each element; the thick contour represents the zero level set. On the right side, a possibly remedy for bridging the gaps in the interface: introduce small interfacial faces aligned with the boundaries of the cells (see centre of blue circles). However, such a technique is overly complicated and intricate, especially in three dimensions; it is not employed in this work. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where, recall, $\mathcal{Q}_p$ is the space of tensor product polynomials used in the dG method. In practice, for rectangular elements, the integral is approximated with a tensor-product Gaussian quadrature with $\lceil \frac{3p+1}{2} \rceil$ nodes per dimension; for curved elements, the precomputed quadrature schemes (see §2.2.2) are reused; in both cases, the L^2 minimisation problem amounts to a symmetric positive definite linear system for the nodal coefficients of u, which is solved with a Cholesky factorisation of the elements mass matrix. In general, for the applications presented in this work involving an evolving level set function, reinitialisation is performed about every 100 time steps—a typical period of time by which physical currents near the interface begin to appreciably shear the level set function. Note that, between reinitialisation steps, ϕ will develop discontinuities in the derivative across the interface, yet high-order accuracy is nevertheless attained as previously remarked.

2.6.3. Elemental to cellular conversion of the level set function

Thus far, we have defined the discrete level set function $\phi \in V_h$ as a piecewise polynomial function to be evolved within a dG method. Consequently, note that ϕ is likely to develop discontinuities (of magnitude comparable to the spatial discretisation error) across element boundaries as it evolves. These discontinuities, if left unaccounted for, leave "gaps" in the implicitly defined interface, as illustrated in Fig. 14. In order to define an implicit mesh using ϕ, which by construction assumed that $\partial\{\phi < 0\} \cap \partial\{\phi > 0\} = \Gamma$, these gaps should be accounted for; one possibility is to somehow bridge the gaps, e.g., by creating small interphase faces that are flat and aligned with the faces of the background quadtree/octree, see, e.g., Fig. 14 right. However, this construction leads to an unnecessarily complex and overly complicated implementation, especially in three dimensions. Another possibility is to, in some sense, leave gaps behind in the interface—however, although the gaps are of size equal to the spatial discretisation error and thus very small, experiments indicated the LDG operators defined in §2.3 are sensitive to gaps in a way that compromises high-order accuracy. The preceding discussion partly motivates the following design goals: after evolving ϕ for one or more time steps, for the sole purpose of defining the implicit mesh, we would like to:

- Remove the discontinuities in ϕ (that are introduced by dG advection), in order to provide a well-defined implicit description of each phase and each interface;
- Straightforwardly compute the volume fractions for the cell merging procedure;
- Employ the high-order quadrature schemes in [29]—which operate with hyperrectangles and are most efficient when applied to polynomial level set functions—to build quadrature schemes for curved elements and non-trivial faces.

All three of these goals can be accomplished by using the high-order accurate distance algorithms discussed in the previous section, as follows. For a two-phase system, for each cell in the quadtree/octree, we define a tensor-product polynomial of one-dimensional degree p (thus coinciding with the degree of the dG polynomial space), as the nodal interpolant (using tensor-product Gauss–Lobatto nodes) of $x \mapsto d_\phi(x)$, where d_ϕ is the signed distance function (computed with input ϕ) as defined in the previous section §2.6.2. This algorithm defines a piecewise polynomial function ψ that (i) preserves the location of the interface with high-order accuracy, and (ii) is continuous throughout the domain.[9] A few remarks are in order:

- The piecewise polynomial function ϕ is referred to as *elemental*, whereas the piecewise polynomial function ψ is referred to as *cellular*. The cellular function converts ϕ, defined on the implicit mesh, to a function defined on the cells of the background quadtree/octree.
- The primary reason for performing the conversion is so that one can construct high-order quadrature schemes for implicitly defined surfaces and volumes. In this work, the algorithms of [29] are used, and these assume that the function implicitly defining the geometry is itself smooth. Consequently, one cannot use ϕ itself (even if its discontinuities across mesh faces were removed), as physical currents generally render ϕ nonsmooth across the interface. Among a variety

[9] Assuming all quadtree/octree cells containing the interface are the same size, the piecewise polynomial nodal interpolant of d_ϕ must necessarily be continuous across all such cells. When the quadtree/octree has regions of different refinement levels, in which some parts of the interface exist in large cells whilst other parts in small cells, it is possible to define ψ in a way that preserves continuity, though we do not consider this further in this work.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

of choices for defining a smooth function with a specific zero level set, the signed distance function is particularly natural.[10]

- The cellular level set function does not replace the elemental level set function: ψ is solely used to define the implicit mesh at every time step, but ϕ itself is left unaltered. This prevents errors from accumulating over many time steps.
- In practice, it is unnecessary to define ψ on *all* cells of the quadtree/octree: instead, one can calculate ψ in a small narrow band on either side of the interface. The methods of §2.6.2—in particular, searching for closest seed points within a given distance threshold as part of the k–d tree mechanism—easily allow for efficient implementation of a narrow banding procedure such as this. Consequently, the computational cost of constructing ψ every time step is negligible: for the applications presented in this work, timing measurements indicate that creating ψ is cheaper than evaluating curved elemental mass matrices, which, in turn, is significantly cheaper than the solution to various PDE problems.

In summary, although at first it may seem undesirable to compute ψ each time step, the combined operation of ϕ and ψ yields several advantages, including: (i) one can advect ϕ by the physical velocity $\mathbf{u}$ using high-order accurate dG methods that have minimal artificial viscosity/numerical dissipation, (ii) maintain high-order accuracy in defining the implicit mesh at every time step, and (iii) increase computational efficiency of the quadrature schemes for implicitly defined surfaces and volumes, as they are most efficient when applied to piecewise polynomial functions.

2.6.4. Transferring state between meshes and preserving mesh topology

To summarise the discussion so far on evolving meshes, we have defined how to evolve the interface (by evolving an elemental piecewise polynomial function ϕ) and how to use the discrete level set function to build an implicit function (denoted ψ in the previous section) suitable for the construction of the implicit mesh in §2.1. In this section, we consider how to incorporate this into a time stepping scheme, focusing on the transference of state (ϕ, $\mathbf{u}$, etc.) between one mesh (at one time step) and the next mesh (at the next time step).

State transference. To illustrate the essential concepts, we consider here a simple first order time stepping scheme, based on forward Euler. Specific applications in part two—see [3]—will consider higher-order accurate time stepping schemes. Abstractly, the time stepping loop consists of:

1. Define, at $t=0$, the initial mesh (denoted $\Omega^{n=0}$) and thus the piecewise polynomial space $V_h^{n=0}$.
2. Initialise $\phi^0 \in V_h^0$ at $t=0$.
3. For $n=0,1,2,\ldots$
   i) Define the velocity $\mathbf{u}^n \in V_h^{d,n}$ at time step n.
   ii) Compute $\phi^{n+1} \in V_h^n$ such that $\frac{1}{\Delta t}(\phi^{n+1}-\phi^n)+(\mathbf{u}^n\cdot\nabla_h^n)\phi^n=0$, where $\mathbf{u}^n\cdot\nabla_h^n$ denotes the discretisation of the advection operator at time step n.
   iii) Use ϕ^{n+1} to build a cellular level set function ψ; use ψ to define the mesh at the next time step, Ω^{n+1}, and thus the dG space V_h^{n+1}.
   iv) Transfer $\phi^{n+1} \in V_h^n$ to the new mesh such that $\phi^{n+1} \in V_h^{n+1}$.
   v) Loop to top.

Fig. 15 motivates the ideas behind steps (ii)–(iv) with a two-dimensional representative example involving two elements on either side of an interface, such that the smoothness of ϕ on either side is clearly different. Step by step, first note that (3ii) computes an updated level set function ϕ^{n+1} at time step $n+1$, but the updated function is nevertheless defined on the old mesh. This is important because the regularity (i.e., the smoothness) of the quantities ϕ^n and $\mathbf{u}^n$ are aligned with the interface at time step n. Thus, $\mathbf{u}^n\cdot\nabla_h^n\phi^n$ can be computed with high-order spatial accuracy on the mesh at time step n. It follows that, using a one-sided Taylor series argument, the interface defined by ϕ^{n+1} is high-order accurate in space and has a temporal local truncation error of size $\mathcal{O}(\Delta t^2)$. The discrete level set function can thus be used to define the mesh at the next time step $n+1$.[11] Having done so, state quantities, including ϕ^{n+1} are then "instantaneously" transferred to the new mesh. This process is referred to as instantaneous because there is no evolution in transferring $\phi^{n+1} \in V_h^n$ to $\phi^{n+1} \in V_h^{n+1}$. Instead, this transference is reflective of the fact that the state quantities at time step $n+1$ are expected to have their smoothness align with the interface at time step $n+1$. Fig. 15 illustrates these ideas: one can observe that $\phi^{n+1} \in V_h^n$ is not high-order accurate on the old mesh, whereas, on the new mesh, $\phi^{n+1} \in V_h^{n+1}$ recovers high-order accuracy because its smoothness conforms to the new interface location. These concepts apply equally to other quantities of state, such as fluid

[10] More precisely, the signed distance function is smooth except for the set of points which do not have a unique closest point. It is generally assumed in this work that the curvature of the interface is resolved, and thus there is at least one cell on both sides of the interface in which the signed distance function is smooth.

[11] In more detail, $\phi^{n+1} \in V_h^n$ is first converted to a cellular level set function ψ using the method described in §2.6.3. This, in turn, uses the high-order accurate distance functions computed by the algorithm in §2.6.2. In that algorithm, the polynomial used to implicitly define the interface is defined by the average of the two elemental polynomials on either side of the interface. This average polynomial, which involves a small amount of extrapolation of the polynomials of $\phi^{n+1} \in V_h^n$, is itself smooth, and accurately defines the interface at time step $n+1$.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

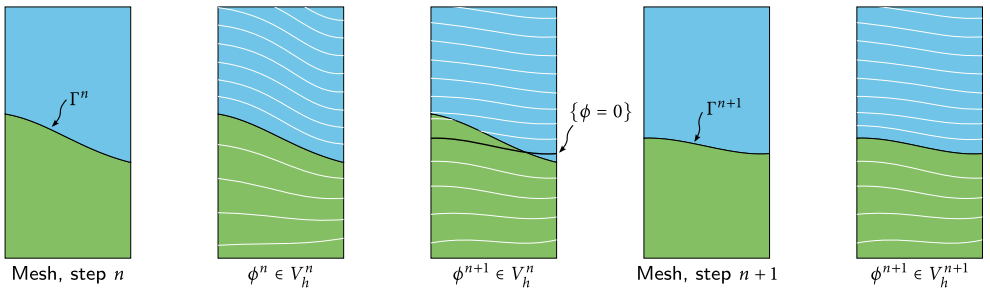

Fig. 15. A schematic of the steps comprising the evolution of a level set function ϕ and the implicitly defined mesh. (a) At time step n, Γ^n separates an element in one phase from an element in a different phase. (b) Contours of a level set function ϕ at time step n; note the change in contour behaviour across the interface, illustrating the property that ϕ may have different smoothness in each phase (e.g., $\nabla\phi$ may exhibit a discontinuity across Γ). (c) After time stepping ϕ for one time step (e.g., by forward Euler), ϕ^{n+1} is, temporarily, defined on the old mesh. The computed interface is accurate, however ϕ^{n+1} develops discontinuities across the (old) interface Γ^n. (d) Using the zero level set of $\phi^{n+1} \in V_h^n$, a new mesh at time step $n+1$ is implicitly defined. (e) By copying the polynomial of ϕ^{n+1} from the old mesh into the new, ϕ^{n+1} is "injected" into the new mesh with the result that $\phi^{n+1} \in V_h^{n+1}$ recovers high-order spatial accuracy.

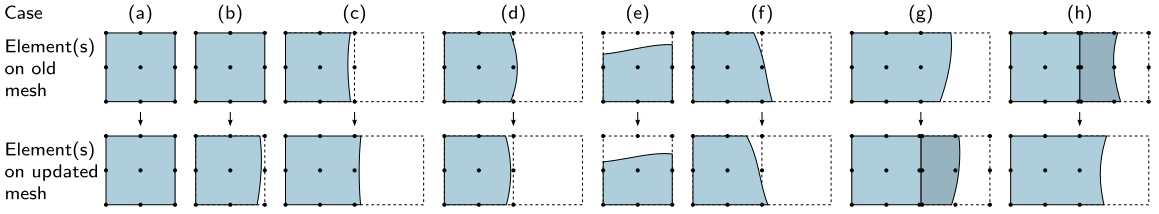

Fig. 16. Representative examples of evolving phase cells and element shapes, demonstrated with a biquadratic nodal basis. (a) A rectangular element unchanging in shape. (b) A rectangular element changing to an unextended curved element. (c) Unextended to extended curved element, involving the creation of a small phase cell. (d) Disappearance of a small phase cell. (e) A large phase cell changing shape. (f) Large and small phase cells changing shape, preserving the same cell merging operation. (g) A small phase cell increases in size to become a large phase cell, thereby creating a new element. (h) A large phase cell decreases in size to become small, thereby destroying an element. In all cases except for (g, h), the values of the degrees of freedom (of $u^{n+1} \in V_h^n$) are directly copied to the new mesh (such that $u^{n+1} \in V_h^{n+1}$), i.e., the polynomial is copied. In case (g), an element is born, and its degrees of freedom are specified by polynomial interpolation. In case (h), an element is destroyed and absorbed into the new element; the polynomial on the new element is defined by an L^2 projection of the piecewise polynomial defined by the old two elements.

velocity: $\mathbf{u}^{n+1} \in V_h^{d,n}$ is not high-order accurate, but when transferred to the new mesh, $\mathbf{u}^{n+1} \in V_h^{d,n+1}$, recovers high-order accuracy.

Consider now how to transfer state defined on an old mesh, Ω^{old}, to a new mesh, Ω^{new}. (Here, "state", denoted u, means a quantity like the level set function or velocity field, understood to be valid at a point in time coinciding with Ω^{new}.) The transference operator is defined in two ways: first schematically, by identifying the different ways elements change shape, and second, in an equivalent way as a form of L^2 projection. Conceptually there are three different possibilities for how the elements have evolved (see also Fig. 16, which illustrates a variety of representative cases):

(a) On all elements with unchanged shape, it is clear how to proceed: simply preserve the coefficients in the dG basis, in precisely the same way as an Eulerian PDE method. (Note that this operation is equivalent to both nodal interpolation and local element-wise L^2 projection.)
(b) For all elements that are determined by the same cell-merging procedure, "inject" the polynomial from the old mesh element E^{old} into the new element E^{new} by preserving its coefficients. This possibility covers the case that an element may grow or shrink slightly due to a moving interface, but whose children phase cells (in the nomenclature of §2.2.5) remain identical. The injection procedure implies that, for all points $x \in E^{\text{new}}$ such that $x \notin E^{\text{old}}$, the value of the state quantity at x is determined by extrapolating the polynomial on E^{old}.
(c) For all other elements, a type of L^2 projection is used which essentially: (i) creates a new element (with its own set of degrees of freedom) whenever a small phase cell on Ω^{old} increases in size to become a large phase cell on Ω^{new}, and/or (ii) absorbs the degrees of freedom of an element on Ω^{old} whenever it was associated with a large phase cell and subsequently shrunk to a small phase cell on Ω^{new}. This L^2 projection procedure is local and utilises a extrapolation similar to (b), and is defined next.

We now state an equivalent mechanism for the transfer operator, covering all of the cases above. Given $u \in V_h^{\text{old}}$ on the old mesh, a piecewise polynomial *cellular* function u_c is defined on the phase cells of the new mesh, as follows. Let $U_i \cap \Omega_j^{\text{new}}$ denote a nonempty phase cell of the new mesh—we consider three cases:

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

- If $U_i \cap \Omega_j^{\text{new}}$ is *entire*, then $U_i \cap \Omega_j^{\text{old}}$ on the old mesh must be *large* or *entire*. (Throughout this section, we assume that the time step Δt is a sufficiently small fraction of the CFL condition, and thus, the interface cannot move more than 25%, say, across a cell in a single time step.) We define u_c on $U_i \cap \Omega_j^{\text{new}}$ as the polynomial coinciding with u restricted to $U_i \cap \Omega_j^{\text{old}}$.
- If $U_i \cap \Omega_j^{\text{new}}$ is *large*, then $U_i \cap \Omega_j^{\text{old}}$ on the old mesh is either *small*, *large*, or *entire*. Again, we define u_c on $U_i \cap \Omega_j^{\text{new}}$ as the polynomial coinciding with u restricted to $U_i \cap \Omega_j^{\text{old}}$.
- If $U_i \cap \Omega_j^{\text{new}}$ is *small*, then $U_i \cap \Omega_j^{\text{old}}$ on the old mesh is either *empty*, *small*, or *large*.
  - If $U_i \cap \Omega_j^{\text{old}}$ on the old mesh is *small* or *large*, we again define u_c on $U_i \cap \Omega_j^{\text{new}}$ as the polynomial coinciding with u restricted to $U_i \cap \Omega_j^{\text{old}}$.
  - On the other hand, if $U_i \cap \Omega_j^{\text{old}}$ on the old mesh is *empty*, then we are in a situation where a small phase cell (on the new mesh) is "born." This new small phase cell has already been merged with a large/entire phase cell on the new mesh (denoted $U_k \cap \Omega_j^{\text{new}}$). Because the interface is assumed to only move a maximum fraction of each cell in each time step, it must be the case that $U_k \cap \Omega_j^{\text{old}}$ must be nonempty on the old mesh. We define in this scenario u_c on $U_i \cap \Omega_j^{\text{new}}$ to be the polynomial extrapolation of u restricted to $U_k \cap \Omega_j^{\text{old}}$.

With the cellular function u_c in hand, we define u on the new mesh Ω^{new} to be the L^2 projection of u_c. Specifically, for every element E of Ω^{new}, E is the union of a large/entire phase cell, plus zero or more small phase cells, $E = \bigcup_{i\in\mathcal{I}} U_i \cap \Omega_{\chi(E)}$, and so

$$u|_E = \operatorname*{arg\,min}_{p\in\mathcal{P}} \sum_{i\in\mathcal{I}} \int_{U_i\cap\Omega^{\text{new}}_{\chi(E)}} (p - u_c)^2,$$

where $\mathcal{P}$ is the space of tensor product polynomials employed by the dG method. The solution of this optimisation problem is straightforward: briefly, it consists of finding $u|_E$ such that $(\sum_{i\in\mathcal{I}} M_i)u|_E = \sum_{i\in\mathcal{I}} M_i u_{c,i}$, where M_i is the mass matrix for the phase cell $U_i \cap \Omega^{\text{new}}_{\chi(E)}$ relative to the nodal basis of E and $u_{c,i}$ is the restriction of u_c to $U_i \cap \Omega^{\text{new}}_{\chi(E)}$ in the nodal basis of E; the resulting linear system is symmetric positive definite and can be solved via Cholesky factorisation.

The above construction effectively absorbs degrees of freedom from the old mesh whenever a large phase cell changes to a small phase cell (see, e.g., Fig. 16h), and creates degrees of freedom on the new mesh whenever a small phase cell grows in size to become a large phase cell (see, e.g., Fig. 16g). Although the description defines the transfer operator in full generality, in practice it is unnecessary to build u_c globally, nor perform the L^2 projection for every element: for the vast majority of elements and time steps, situations (a) and (b) above apply and it is simple to copy/inject the polynomials from one mesh to the next; when situation (c) occurs, the function u_c can be built locally and the local L^2 projection efficiently computed.

Preserving mesh topology. To conclude the discussion on remeshing, we consider a simple improvement to the cell merging algorithm used in the construction of implicitly defined meshes. The modification minimises errors introduced by state transference, increases remeshing efficiency, and forms a key component in accurately computing temporal derivatives within the interfacial gauge methods considered in part two [3].

As motivation, note that the state transfer operator is particularly simple when there is a one-to-one correspondence between elements on the old mesh and those on the new. Specifically, when only situation (a) or (b) above occurs, there is natural mapping from $u \in V_h^{\text{old}}$ to $u \in V_h^{\text{new}}$ which essentially leaves the degrees of freedoms unaltered. Thus, it would be beneficial to maximise the chances of this event, and this can be achieved by utilising "fuzzy" thresholds in the cell merging procedure of §2.1. To illustrate, consider a case where, for example, a phase cell has a volume fraction of 39% at one time step and 41% at the next. If a hard threshold of 40% (as used in §2.1) is used, then the two time steps would involve different cell merging procedures. However, the phase cell is essentially the same for both time steps, and so in this regard a hard threshold is unnecessary. In constructing the new mesh, one can instead classify (as small or large) the new phase cell in the same way as the corresponding phase cell on the old mesh, and identify when doing so is inadvisable (i.e., the interface has moved too far) with a fuzzy threshold. Specifically, let $\theta = |U_i \cap \Omega_j^{\text{new}}|/|U_i|$ denote the volume fraction of the new phase cell, then:

- If the old phase cell is *small* and $\theta > 0$, then mark the new phase cell as *small*. If $\theta > 0.45$, then set a flag to true.
- If the old phase cell is *large* and $\theta \leq 0.4$, then mark the new phase cell as *large*. If $\theta < 0.35$, then set a flag to true.

In addition, in assigning small phase cells to parent phase cells as part of the cell merging procedure, when possible, the small phase cells of the new mesh are assigned to the same parents as were used on the old mesh. Combined, these alterations redefine the construction of the implicit mesh in §2.1 by forcing the cell classifications to be almost[12] the same.

[12] Large cells are permitted to change to entire cells and vice versa, while small cells are permitted to appear and disappear.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

Consequently, a one-to-one correspondence exists between elements on the old mesh and the new. And, except for the creation or destruction of small interphase/intraphase faces associated with the emergence or disappearance of small cells, the topology of the two meshes is the same (i.e., the topology of the graph with elements representing nodes and faces between elements representing edges). This procedure of using fuzzy thresholds—to match the construction of an implicit mesh with an existing mesh—is referred to as "preserving the mesh topology."

The role of the flag in the above is to ascertain whether the interface has moved too far, e.g., a small phase cell really should be classified as large, and vice versa. (In the above, this is controlled with a ± 0.05 threshold, a user-chosen parameter of reasonable size so as to be neither too strict nor too relaxed.) This flag is used to trigger a remeshing operation in which the new implicitly defined mesh is constructed without any knowledge of an existing mesh, i.e., a 40% volume fraction threshold is used throughout. The applications in part two—see [3]—describe further how this trigger is used.

For the vast majority of time steps, the topology of the mesh can be preserved in this way. Thus, the L^2 projection procedure described previously is, on the whole, executed relatively infrequently. This minimises errors associated with the L^2 projection (which themselves are uncorrelated with Δt) from accumulating unnecessarily often. For example, the procedure automatically stabilises cases in which cell volume fractions oscillate between slightly less than 40% and slightly greater than 40%, without having to frequently create and destroy elements in the process.

Summary. In this section, we discussed and defined operators which transfer state between evolving meshes in a time stepping algorithm. A simple modification to the construction of implicitly defined meshes, using fuzzy thresholds, allows the state to be transferred very simply, as frequently as possible, i.e., by copying the nodal values of the piecewise polynomial functions. In this way, the remeshing procedure and time stepping methods are essentially Eulerian in character (compared to, say, Lagrangian). The Eulerian formulation provides a variety of advantages, including benefiting from the regularisation properties of level set methods [8], relatively simple time stepping algorithms, and straightforward calculation of temporal derivatives, $\frac{\partial}{\partial t}$, as seen in the applications considered in part two [3].

2.6.5. Briefly on conservation properties

In the context of evolving interface problems, the described implicit mesh dG framework is not, in general, conservative. In other words, if the governing physical equations of motion conserve some quantity such as mass or momentum, the same quantity in the discrete setting may not be exactly conserved by the discrete method. There are two main causes for this. First, for problems in which one may expect conservation of volume of each phase (e.g., in incompressible multiphase fluid flow), neither the evolution of the level set function ϕ (§2.6.1) nor the element-to-cellular conversion of ϕ to ψ (§2.6.3) is expected to conserve the volume of each phase to machine precision. Second, owing to the polynomial injection operation in the state transference procedure (§2.6.4), transferring state between one mesh and the next involves a small amount of extrapolation of polynomial field values whenever an element grows in size. Thus, conservation of discrete quantities is unlikely to hold in practice for moving interface problems. Nevertheless, note that these discrepancies in conservation generally relate to high-order spatial truncation errors. Note also that these observations apply only to that part of the mesh with changing element shape—for a static implicit mesh, or away from a moving interface, the usual conservation properties of dG methods hold, e.g., quantities evolved by a hyperbolic conservation law are conserved locally by the discrete method.

Notwithstanding the above, it is important to note the implicit mesh dG methodology nevertheless recovers conserved quantities to high-order accuracy whenever the dynamics are sufficiently resolved. For example, a two-phase surface tension dynamics example considered in part two [3]—which considers a soap bubble oscillation problem on a moderately refined mesh, with a Reynolds number in the tens of thousands, and makes no attempt to conserve mass of the gas inside and outside the bubble in the discrete setting—conserves mass to one part in 8000. Under-resolved cases, e.g., dispersal of liquid droplets with diameter comparable to or smaller than the typical element size, are not considered as part of the present work. In under-resolved cases, the construction of the implicit mesh and the transference of state may need to be augmented with other strategies to assist with any desired conservation properties; possibilities may include, for example, adopting ideas from volume of fluid methods [13], which can conserve fluid mass to machine precision, to develop a hybrid interface tracking algorithm, see, e.g., [66,67].

2.6.6. Adaptive mesh refinement

To conclude the development of the implicit mesh dG framework, we briefly discuss the possibility of adaptive mesh refinement (AMR). Owing to the flexibility discontinuous Galerkin methods have in using unstructured meshes, it is relatively straightforward to implement a form of AMR within implicitly defined meshes, essentially by refining the background quadtree/octree. In this work, sophisticated refinement strategies based on locally computed error metrics have not been considered; instead, either (i) no refinement, (ii) static, "hand-crafted" refinement, or (iii) refinement based on proximity to the evolving interface is used. In each case, the background quadtree/octree is refined based on a length scale function $h : \Omega \to \mathbb{R}$, such that cells within the quadtree/octree halt subdivision as soon as their width is smaller than $h(x_c)h_0$, where x_c is the centroid of the cell and the constant h_0 is a user-supplied length scale. In case (i), $h \equiv 1$; in case (ii), h varies in space but is fixed throughout the computation, and in case (iii), h is periodically updated as a function of the distance to the interface. For an example, see Fig. 13 of part two [3]. After adaptively refining the background quadtree/octree, the cell merging procedure and thus the construction of the implicit mesh proceeds unaltered. In particular, we make note of one aspect: if the quadtree/octree refinement is such that one part of the interface has small cells (denoted region "A" in the following), whereas other parts of the interface have larger cells (region "B"), then the mesh will change grade somewhere

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

on the interface; with ad hoc merging strategies, the cell merging procedure may produce unideal results, as small phase cells from region B could be merged with large phase cells in region A in such a way that highly skewed elements are formed. A simple strategy—whereby small phase cells (from any region) are merged with neighbouring phase cells based on the largest volume *fraction* (i.e., $|U_i \cap \Omega_j|/|U_i|$)—avoids this issue and results in more ideal element shapes. This strategy, being based on volume fractions rather than the volume of each phase cell, is that which is already described by the algorithms detailed in §2.1. More sophisticated cell merging strategies may be possible, particularly in the case of a mesh with rapidly varying levels of refinement, however this is not pursued further here; see also, e.g., [1,30,31] for considerations of other cell merging procedures.

3. Interim remarks

In the presentation so far, we have discussed an implicit mesh dG framework based around the idea of an implicitly defined mesh, having in mind the goal of computing high-order accurate interface dynamics. In the second part of this work—see [3]—the 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. For each of these applications, the equations of motion are stated, suitable time stepping methods are designed, and convergence analyses are performed to verify high-order accuracy. Part two also discusses how to adapt the implicit mesh dG framework to axisymmetric modelling, and concludes with a summary of the main ideas underlying the methodologies developed in the present paper as well as part two.

Acknowledgements

I thank Per-Olof Persson for many interesting discussions regarding the design of discontinuous Galerkin methods, and in particular, for initially pointing out that among a variety of possible discretisations of elliptic PDE problems, the local discontinuous Galerkin method combined with one-sided numerical fluxes (as used in this work) has the useful property that the projection operator is discretely idempotent (ignoring penalty parameters). 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.

References

[1] A. Johansson, M.G. Larson, A high order discontinuous Galerkin Nitsche method for elliptic problems with fictitious boundary, Numer. Math. 123 (4) (2013) 607–628, http://dx.doi.org/10.1007/s00211-012-0497-1.
[2] R. Saye, Interfacial gauge methods for incompressible fluid dynamics, Sci. Adv. 2 (6) (2016), http://dx.doi.org/10.1126/sciadv.1501869.
[3] 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 II, J. Comput. Phys. (2017), http://dx.doi.org/10.1016/j.jcp.2017.05.003.
[4] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517, http://dx.doi.org/10.1017/S0962492902000077.
[5] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261, http://dx.doi.org/10.1146/annurev.fluid.37.061903.175743.
[6] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (4) (1994) 1019–1044, http://dx.doi.org/10.1137/0731054.
[7] Z. Li, K. Ito, The Immersed Interface Method: Numerical Solutions of PDES Involving Interfaces and Irregular Domains, Society for Industrial and Applied Mathematics, 2006.
[8] 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.
[9] 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.
[10] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1) (1994) 146–159, http://dx.doi.org/10.1006/jcph.1994.1155.
[11] J.A. Sethian, P. Smereka, Level set methods for fluid interfaces, Annu. Rev. Fluid Mech. 35 (2003) 341–372, http://dx.doi.org/10.1146/annurev.fluid.35.101101.161105.
[12] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the Ghost Fluid Method), J. Comput. Phys. 152 (2) (1999) 457–492, http://dx.doi.org/10.1006/jcph.1999.6236.
[13] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225, http://dx.doi.org/10.1016/0021-9991(81)90145-5.
[14] V.-T. Nguyen, J. Peraire, B.C. Khoo, P.-O. Persson, A discontinuous Galerkin front tracking method for two-phase flows with surface tension, Comput. Fluids 39 (1) (2010) 1–14, http://dx.doi.org/10.1016/j.compfluid.2009.06.007.
[15] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, Int. J. Numer. Methods Eng. 46 (1) (1999) 131–150, http://dx.doi.org/10.1002/(SICI)1097-0207(19990910)46:1<131::AID-NME726>3.0.CO;2-J.
[16] F. Alauzet, B. Fabrèges, M.A. Fernández, M. Landajuela, Nitsche-XFEM for the coupling of an incompressible fluid with immersed thin-walled structures, Comput. Methods Appl. Mech. Eng. 301 (2016) 300–335, http://dx.doi.org/10.1016/j.cma.2015.12.015.
[17] A.J. Lew, G.C. Buscaglia, A discontinuous-Galerkin-based immersed boundary method, Int. J. Numer. Methods Eng. 76 (4) (2008) 427–454, http://dx.doi.org/10.1002/nme.2312.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

[18] P. Bastian, C. Engwer, An unfitted finite element method using discontinuous Galerkin, Int. J. Numer. Methods Eng. 79 (12) (2009) 1557–1576, http://dx.doi.org/10.1002/nme.2631.
[19] F. Heimann, C. Engwer, O. Ippisch, P. Bastian, An unfitted interior penalty discontinuous Galerkin method for incompressible Navier–Stokes two-phase flow, Int. J. Numer. Methods Fluids 71 (3) (2013) 269–293, http://dx.doi.org/10.1002/fld.3653.
[20] D. Devendran, D.T. Graves, H. Johansen, A higher-order finite-volume discretization method for Poisson's equation in cut cell geometries, ArXiv e-prints, Nov. 2014, arXiv:1411.4283.
[21] B. Muralidharan, S. Menon, A high-order adaptive Cartesian cut-cell method for simulation of compressible viscous flow over immersed bodies, J. Comput. Phys. 321 (2016) 342–368, http://dx.doi.org/10.1016/j.jcp.2016.05.050.
[22] J. Nitsche, Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind, Abh. Math. Semin. Univ. Hamb. 36 (1) (1971) 9–15, http://dx.doi.org/10.1007/BF02995904.
[23] S. Sticko, G. Kreiss, A stabilized Nitsche cut element method for the wave equation, Comput. Methods Appl. Mech. Eng. 309 (2016) 364–387, http://dx.doi.org/10.1016/j.cma.2016.06.001.
[24] E. Burman, P. Hansbo, M.G. Larson, A. Massing, A cut discontinuous Galerkin method for the Laplace–Beltrami operator, IMA J. Numer. Anal. 37 (1) (2017) 138–169, http://dx.doi.org/10.1093/imanum/drv068.
[25] P. Hansbo, M.G. Larson, S. Zahedi, A cut finite element method for a Stokes interface problem, Appl. Numer. Math. 85 (2014) 90–114, http://dx.doi.org/10.1016/j.apnum.2014.06.009.
[26] A. Johansson, M. Garzon, J.A. Sethian, A three-dimensional coupled Nitsche and level set method for electrohydrodynamic potential flows in moving domains, J. Comput. Phys. 309 (2016) 88–111, http://dx.doi.org/10.1016/j.jcp.2015.12.026.
[27] E. Burman, S. Claus, P. Hansbo, M.G. Larson, A. Massing, CutFEM: discretizing geometry and partial differential equations, Int. J. Numer. Methods Eng. 104 (7) (2015) 472–501, http://dx.doi.org/10.1002/nme.4823.
[28] J.S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Springer, 2008.
[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] J. Hunt, An Adaptive 3D Cartesian Approach for the Parallel Computation of Inviscid Flow About Static and Dynamic Configurations, PhD thesis, University of Michigan, 2004.
[31] A. Fröhlcke, A Boundary Conformal Discontinuous Galerkin Method for Electromagnetic Field Problems on Cartesian Grids, PhD thesis, Technische Universität, Darmstadt, 2013.
[32] B. Müller, S. Krämer-Eis, F. Kummer, M. Oberlack, A high-order discontinuous Galerkin method for compressible flows with immersed boundaries, Int. J. Numer. Methods Eng. 110 (1) (2017) 3–30, http://dx.doi.org/10.1002/nme.5343.
[33] G. Brandstetter, S. Govindjee, A high-order immersed boundary discontinuous-Galerkin method for Poisson's equation with discontinuous coefficients and singular sources, Int. J. Numer. Methods Eng. 101 (11) (2015) 847–869, http://dx.doi.org/10.1002/nme.4835.
[34] F. Kummer, Extended discontinuous Galerkin methods for two-phase flows: the spatial discretization, Int. J. Numer. Methods Eng. 109 (2) (2017) 259–289, http://dx.doi.org/10.1002/nme.5288.
[35] B. Müller, F. Kummer, M. Oberlack, Highly accurate surface and volume integration on implicit domains by means of moment-fitting, Int. J. Numer. Methods Eng. 96 (8) (2013) 512–528, http://dx.doi.org/10.1002/nme.4569.
[36] P.F. Antonietti, A. Cangiani, J. Collis, Z. Dong, E.H. Georgoulis, S. Giani, P. Houston, Review of discontinuous Galerkin finite element methods for partial differential equations on complicated domains, in: Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin, Heidelberg, 2015, pp. 1–28.
[37] 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.
[38] 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.
[39] 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.
[40] 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.
[41] 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.
[42] 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.
[43] 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.
[44] 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.
[45] 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.
[46] M. Herrmann, A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids, J. Comput. Phys. 227 (4) (2008) 2674–2706, http://dx.doi.org/10.1016/j.jcp.2007.11.002.
[47] V.I. Oseledets. On a new way of writing the Navier–Stokes equation. The Hamiltonian formalism. Russ. Math. Surv. 44 (3) (1989) 210–211 [translated from Comm. Moscow Math. Soc. 1988]. http://dx.doi.org/10.1070/RM1989v044n03ABEH002122.
[48] P.H. Roberts, A Hamiltonian theory for weakly interacting vortices, Mathematika 19 (2) (1972) 169–179, http://dx.doi.org/10.1112/S0025579300005611.
[49] R.I. Saye, J.A. Sethian, The Voronoi implicit interface method for computing multiphase physics, Proc. Natl. Acad. Sci. USA 108 (49) (2011) 19498–19503, http://dx.doi.org/10.1073/pnas.1111557108.
[50] W.E. Lorensen, H.E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, Comput. Graph. 21 (4) (1987) 163–169, http://dx.doi.org/10.1145/37402.37422.
[51] A. Guéziec, R. Hummel, Exploiting triangulated surface extraction using tetrahedral decomposition, IEEE Trans. Vis. Comput. Graph. 1 (4) (1995) 328–342, http://dx.doi.org/10.1109/2945.485620.
[52] B.A. Payne, A.W. Toga, Surface mapping brain function on 3D models, IEEE Comput. Graph. Appl. 10 (5) (1990) 33–41, http://dx.doi.org/10.1109/38.59034.
[53] S.L. Chan, E.O. Purisima, A new tetrahedral tesselation scheme for isosurface generation, Comput. Graph. 22 (1) (1998) 83–90, http://dx.doi.org/10.1016/S0097-8493(97)00085-X.
[54] 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.
[55] P. Castillo, An optimal estimate for the local discontinuous Galerkin method, in: B. Cockburn, G.E. Karniadakis, C.-W. Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, Springer, Berlin, Heidelberg, 2000, pp. 285–290.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076

[56] B. Cockburn, G. Kanschat, I. Perugia, D. Schötzau, Superconvergence of the local discontinuous Galerkin method for elliptic problems on Cartesian grids, SIAM J. Numer. Anal. 39 (1) (2001) 264–285, http://dx.doi.org/10.1137/S0036142900371544.
[57] W.L. Briggs, V.E. Henson, S.F. McCormick, A Multigrid Tutorial, Second edition, Society for Industrial and Applied Mathematics, 2000.
[58] M. Adams, M. Brezina, J. Hu, R. Tuminaro, Parallel multigrid smoothing: polynomial versus Gauss–Seidel, J. Comput. Phys. 188 (2) (2003) 593–610, http://dx.doi.org/10.1016/S0021-9991(03)00194-3.
[59] M.F. Adams, A distributed memory unstructured Gauss–Seidel algorithm for multigrid smoothers, in: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing, ACM, New York, NY, USA, 2001, pp. 1–18.
[60] A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations, J. Comput. Phys. 142 (1) (1998) 1–46, http://dx.doi.org/10.1006/jcph.1998.5890.
[61] 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.
[62] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, 2003.
[63] 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.
[64] D. Adalsteinsson, J.A. Sethian, The fast construction of extension velocities in level set methods, J. Comput. Phys. 148 (1) (1999) 2–22, http://dx.doi.org/10.1006/jcph.1998.6090.
[65] 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.
[66] M. Sussman, E.G. Puckett, A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2) (2000) 301–337, http://dx.doi.org/10.1006/jcph.2000.6537.
[67] M. Owkes, O. Desjardins, A computational framework for conservative, three-dimensional, unsplit, geometric transport with application to the volume-of-fluid (VOF) method, J. Comput. Phys. 270 (2014) 587–612, http://dx.doi.org/10.1016/j.jcp.2014.04.022.

Please cite this article in press as: 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. (2017), http://dx.doi.org/10.1016/j.jcp.2017.04.076