INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2017; **109**:259–289
Published online 22 June 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.5288

Extended discontinuous Galerkin methods for two-phase flows: the spatial discretization

Florian Kummer[1,2,*,†]

[1]*Fluid Dynamics, Technische Universität Darmstadt, Otto-Berndt-Str. 2, D-64287 Darmstadt, Germany*
[2]*Graduate School Computational Engineering, Technische Universität Darmstadt, Dolivostr. 15, D-64293 Darmstadt, Germany*

SUMMARY

This work discusses a discontinuous Galerkin (DG) discretization for two-phase flows. The fluid interface is represented by a level set, and the DG approximation space is adapted such that jumps and kinks in pressure and velocity fields can be approximated sharply. This adaption of the original DG space, which can be performed 'on-the-fly' for arbitrary interface shapes, is referred to as extended discontinuous Galerkin. By combining this ansatz with a special quadrature technique, one can regain spectral convergence properties for low-regularity solutions, which is demonstrated by numerical examples. This work focuses on the aspects of spatial discretization, and special emphasis is devoted on how to overcome problems related to quadrature, small cut cells, and condition number of linear systems. Temporal discretization will be discussed in future works. Copyright © 2016 John Wiley & Sons, Ltd.

Received 22 April 2015; Revised 17 March 2016; Accepted 12 April 2016

KEY WORDS: discontinuous Galerkin; extended/unfitted DG; XDG; XFEM; quadrature; cut cell; multiphase flows; level set

1. MOTIVATION AND OBJECTIVES

Over the past decades, numerous numerical methods for solving partial differential equations with a high order of accuracy have been developed. This means that the error of the numerical solution should asymptotically behave as h^k, where h denotes a characteristic length scale of the computational grid and k denotes the convergence order. In fluid dynamics, an order of at least $k = 2$ is widely accepted as a minimum requirement for acceptable accuracy. However, common to almost all of these methods is that their convergence order usually degenerates for low-regularity solutions, which, for example, occur in two-phase problems. For material interfaces, the velocity field contains a kink (i.e., a jump of the first derivative) at the fluid interface, while the pressure field contains a jump due to surface tension effects.

Two major directions in the numerical treatment of two-phase flows can be identified. At first, approaches regularize jumps in density and viscosity at the interface, so that the variations can be resolved on a given grid. One example for such an approach is the conserved level set method by Olssen and Kreis [1]. In this work, we focus on a second class of methods that employ an adaption of the approximation with respect to the fluid interface. The motivation behind this is that spectral convergence can still be observed for low-regularity problems if the grid is chosen to be conformal with the position of the discontinuity.

The idea of adapting a finite element method (FEM) to allow jumps in parameters can be traced back to the 1970s, where Poisson problems with a jump in the diffusion parameter along a smooth

*Correspondence to: Florian Kummer, TU Darmstadt, Otto-Berndt-Str. 2, D-64287 Darmstadt, Germany.
†E-mail: kummer@fdy.tu-darmstadt.de

interface were investigated by Babuška [2] as well as by Barrett and Elliott [3]. For the mathematical properties, that is, convergence rate and spectral properties of the matrix, the technique of interface representation should have no effect. However, especially for moving interfaces, one typically wants to avoid repetitive remeshing, leading to so-called extended finite element methods (XFEM). A very prominent work in this field, although in the domain of solid mechanics, was presented by Moës *et al.* [4].

The present work is based on the discontinuous Galerkin (DG) method. In cells that are cut by the fluid interface, the standard DG space is extended in order to provide separate degrees of freedom (DOF) for both phases. One difficulty is that one has to integrate accurately over an interface that is only known implicitly and can have an almost arbitrary shape. This is solved by a special procedure for numerical integration, the so-called hierarchical-moment-fitting (HMF) [5] that eliminates the need for reconstructing the interface. On this basis, we demonstrate a DG method for which we can experimentally show a convergence order of approximately $k + 1$ for velocity and k for pressure, where k denotes the total polynomial degree of the velocity approximation. Many names for this idea seem at hand, for example, cut-cell DG or unfitted DG [6]. In reminiscence of XFEM [4], we prefer the name extended discontinuous Galerkin (XDG). While the idea seems simple at first glance, it introduces a bunch of technical difficulties, some of which are addressed in this work.

This work represents the second paper in a series of publications on the road to a fully transient, 3D XDG-multiphase solver. Within the first work [5], the HMF procedure was developed. Because an XDG method like the one proposed in this work can only be 'h^{k+1} - accurate' if the quadrature also is sufficiently accurate, this is a necessary prerequisite for this work. For the actual work, one simplification we make is to consider only polynomial level set functions in $\mathcal{C}^{\infty}(\Omega)$. Such a choice allows us to evaluate the curvature of the fluid interface, which is required to describe the pressure jump due to surface tension effects, up to machine accuracy. This simplification obviously does not hold anymore in transient computations, where the level set function is the result of some time evolution scheme. Therefore, in the third publication [7], the issue of curvature evaluation that is required to describe the pressure jump due to surface tension effects is addressed. Future work will discuss the extension to transient simulations. This is indeed not trivial, because, for example, also the XDG space depends on time as soon as the interface moves. Furthermore, the issue of efficient linear solvers for the saddle point problems formulated herein has to be discussed.

1.1. State of the art

Moës *et al.* introduced the XFEM [4] to simulate cracking phenomena in solid mechanics. Those ideas were extended to time-dependent problems by Chessa and Belytschko [8]. In a fluid dynamics context, this approach was first used by Groß and Reusken [9] who also provided an error analysis [10]. In those works, only the pressure space is extended, while for velocity, still a classical FEM space is used. With the so-called intrinsic XFEM presented by Fries [11], it became possible to represent also kinks in the velocity field. This work was later extended by Cheng and Fries [12] and Sauerland and Fries [13].

Recently, Hansbo *et al.* [14] presented a Nitsche FEM for steady Stokes problems, based on so-called P1-iso-P2 elements. This means linear continuous polynomials for pressure and velocity, where the velocity is defined on a uniformly refined grid. A main ingredient of this work is the 'ghost penalty' stabilization presented by Burman [15].

The works cited so far all represent extensions of the FEM. Our developments, however, are based on the DG method, first introduced by Reed and Hill [16] to model neutron transport problems. Although being the first DG method, there are no many connections to nowadays DG approaches. A review on the application of DG methods to incompressible single-phase flows can, for example, be found in the book of Di Pietro and Ern [17].

Whether DG discretizations are actually a good choice over continuous gradient (CG) methods [18] for fluid dynamic problems is an actual dispute. For incompressible flows, one can formulate DG methods that use equal polynomial degrees for velocity and pressure approximation, if proper stabilization techniques for the pressure are used [17]. Furthermore, they are more flexible with respect to grid refinement, because the method can be formulated with arbitrarily 'hanging'

DOI: 10.1002/nme

nodes. Also on the implementation side, DG methods offer some advantages: Sparse matrices have a block structure, and cells communicate only with their direct neighbors. Therefore, their sparse matrix structure is significantly simpler than those of CG methods. On the other hand, DG methods have a higher number of DOFs for the same grid and interpolation order, in comparison with CG methods. DG methods may overcome this issue by means of hybridization, leading to so-called hybrid discontinuous Galerkin (HDG) methods [19], which drastically reduce the number of globally coupled DOFs. Comparisons of HDG against CG methods were even performed for elliptic problems like the Poisson equation with constant coefficients [20], where the use of CG should be advantageous. Further comparisons in terms of DOFs and number-of-operations were presented by Huerta *et al.* [21, 22]. All these works came to the conclusion that HDG methods are at least not severely inferior to CG methods. Taking into account the aforementioned advantages of DG methods, their application to flow problems seems promising.

The first actual extended/unfitted/cut-cell DG method was presented by Bastian and Engwer [6] in order to model flows in porous media. Later, those approaches were extended to multiphase flows by Heimann *et al.* [23]. An intrinsic problem of all extension approaches are issues related to small cut cells – also large parts of our work presented here are concerned with the treatment of these issues. While our main tool to address these problems is the cell agglomeration procedure, one could also use a mean operator weighted by cell volume ratios, instead of the $1/2$-weighted one used by us (Equation 9), in order to improve the properties of the linear system, as demonstrated by Massjung [24] for the Poisson equation. Obviously, there is a connection to Nitsche methods, where a part of the ansatz space is discontinuous – usually at some interface, while the complement of the ansatz space is continuous. Also these kind of methods, where a background grid is overlaid with an arbitrary interface, require special treatments for small cut cells. In this context, one should mention fictitious domain methods (e.g., [25, 26]).

For an extensive overview on quadrature methods for level sets respectively cut cells, we refer to our original work on HMF [5]. A rather new, alternative approach to quadrature on cut cells was proposed by Saye [27], who also demonstrated an XDG method for a Poisson problem with a jump at the interface.

2. THE CONTINUOUS SETTING

In order to formalize the two-phase flow setting, we define the following disjoint partitioning of the computational domain $\Omega \subset \mathbb{R}^2$:

$$\Omega = \mathfrak{A} \,\dot\cup\, \mathfrak{I} \,\dot\cup\, \mathfrak{B}. \tag{1}$$

In this work, we restrict ourselves to two-dimensional settings. Here, the sets $\mathfrak{A}$ and $\mathfrak{B}$ denote the individual fluid phases. The density ρ and the viscosity μ are piecewise constant in $\mathfrak{A}$ and $\mathfrak{B}$:

$$\rho(\vec{x}) = \begin{cases} \rho_{\mathfrak{A}} & \text{for } \vec{x} \in \mathfrak{A} \\ \rho_{\mathfrak{B}} & \text{for } \vec{x} \in \mathfrak{B} \end{cases} \quad \text{and} \quad \mu(\vec{x}) = \begin{cases} \mu_{\mathfrak{A}} & \text{for } \vec{x} \in \mathfrak{A} \\ \mu_{\mathfrak{B}} & \text{for } \vec{x} \in \mathfrak{B} \end{cases}. \tag{2}$$

The set $\mathfrak{I}$ denotes the fluid interface. In mathematical terms, we assume it to be a one-dimensional manifold that has at least a continuous and globally bounded curvature. The two-fluid formulation of the incompressible Navier–Stokes equation is given by the bulk equations in $\Omega \setminus \mathfrak{I}$,

$$\operatorname{div}(\rho\vec{u} \otimes \vec{u}) + \nabla p = \operatorname{div}\left(\mu\left(\nabla\vec{u} + \left(\nabla\vec{u}\right)^T\right)\right) - \vec{F}, \tag{3}$$

$$\operatorname{div}(\vec{u}) = 0, \tag{4}$$

and the jump conditions at the interface $\mathfrak{I}$,

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

$$\left[\!\left[p\vec{n}_{\mathfrak{I}} - \mu\left(\nabla\vec{u} + \left(\nabla\vec{u}\right)^T\right)\vec{n}_{\mathfrak{I}}\right]\!\right] = \sigma\kappa\vec{n}_{\mathfrak{I}}, \tag{6}$$

DOI: 10.1002/nme

where σ denotes the surface tension, κ the mean curvature of $\mathfrak{I}$, and $\vec{n}_\mathfrak{I}$ a normal field on the interface $\mathfrak{I}$, with an orientation that 'points from $\mathfrak{A}$ to $\mathfrak{B}$.' For the definition of the jump operator $[\![-]\!]$, see Equation (8). Furthermore, one has to specify boundary conditions

$$\vec{u} = \vec{u}_\mathrm{D} \text{ on } \Gamma_\mathrm{D} \text{ and } \mu\left(\nabla\vec{u} + \nabla\vec{u}^T\right)\cdot\vec{n}_{\partial\Omega} - p\vec{n}_{\partial\Omega} = 0 \text{ on } \Gamma_\mathrm{N}, \tag{7}$$

where $\partial\Omega = \Gamma_\mathrm{D} \dot\cup \Gamma_\mathrm{N}$ denotes a disjoint decomposition of the boundary into a Dirichlet and a Neumann boundary region. The right-hand side of the momentum Equation (3) can obviously be simplified, because given that μ is piecewise constant and $\mathrm{div}\left(\vec{u}\right) = 0$, one obtains

$$\mathrm{div}\left(\mu\left(\nabla\vec{u} + \left(\nabla\vec{u}\right)^T\right)\right) = \mu\Delta\vec{u}.$$

In this work, however, we opted to discretize form (3) of the momentum equation, so that the same formulation for bulk and interface terms (Equation 6) can be used.

Because we focus only on the spatial discretization, we do not consider a temporal evolution of the phases. Note that, from a physical point of view, the presented setting is somewhat contradictory: If the interface does not move in time, the velocity of the fluid in normal direction to the interface needs to be zero $\left(\vec{u}\cdot\vec{n}_\mathfrak{I} = 0\right)$. From a mathematical point of view, however, we only enforce $[\![\vec{u}]\!] = 0$, but not that the interface has to move with the same speed as the flow. This may be interpreted as a singular mass source of the form $\Delta_\mathfrak{I}[\![\rho\vec{u}\cdot\vec{n}_\mathfrak{I}]\!]$, where $\Delta_\mathfrak{I}$ denotes a delta distribution at the interface. Although being unphysical, this setting is still suitable to study convergence order as well as stability issues of the spatial discretization.

3. THE EXTENDED DISCONTINUOUS GALERKIN DISCRETIZATION OF THE TWO-PHASE PROBLEM

In this section, a formal definition of the proposed discretization of the two-phase problem is given. First, we recall some basic DG-related definitions (Section 3.1); second, we introduce the extended DG space (Section 3.2); and finally, we apply this framework onto the two-phase problem (Section 3.3).

3.1. A few definitions

While at least trying to keep the amount of Bourbakism in this work as low as possible, we have to recall some basic definitions and notations that form the DG framework. These are fairly standard and can be found in similar form in many textbooks [17, 28].

Definition 1 (basic notations)
We define

- the computational domain: $\Omega \subset \mathbb{R}^2$ that must be polygonal and simply connected;
- the numerical grid: $\mathfrak{K}_h = \{K_1, \ldots, K_J\}$, with h being the maximum diameter of all cells K_j. The cells cover the whole domain $\left(\overline{\Omega} = \bigcup_j \overline{K_j}\right)$ but do not overlap $\left(\int_{K_j\cap K_l} 1\mathrm{dV} = 0 \text{ for } l \neq j\right)$. We restrict ourself to non-curved grids, that is, each cell K_j can be described as the image of a reference cell K_ref under an affine-linear mapping $T_j : \mathbb{R}^2 \to \mathbb{R}^2$, that is, $T_j(K_\mathrm{ref}) = K_j$;
- the set containing all edges of the grid: $\Gamma := \bigcup_j \partial K_j$. Furthermore, the set of all internal edges: $\Gamma_\mathrm{int} := \Gamma \setminus \partial\Omega$;
- a normal field $\vec{n}_\Gamma$ on Γ. On $\partial\Omega$, it represents an outer normal, that is, on $\partial\Omega$, $\vec{n}_\Gamma = \vec{n}_{\partial\Omega}$. By $\vec{n}_{\mathfrak{I},\Gamma}$, we denote a normal field that is equal to $\vec{n}_\Gamma$ on Γ and equal to $\vec{n}_\mathfrak{I}$ on $\mathfrak{I}$;
- the jump and the average value operator on the sets Γ and $\mathfrak{I}$: for some $u \in \mathcal{C}^0(\Omega \setminus \Gamma_\mathrm{int} \setminus \mathfrak{I})$ and $\vec{x} \in \Gamma_\mathrm{int} \cup \mathfrak{I}$,

DOI: 10.1002/nme

$$[\![u]\!](\vec{x}) := \lim_{\xi \searrow 0} \left(u\left(\vec{x} + \xi \vec{n}_{\mathfrak{I},\Gamma}\right) - u\left(\vec{x} - \xi \vec{n}_{\mathfrak{I},\Gamma}\right)\right), \tag{8}$$

$$\{\!\{u\}\!\}(\vec{x}) := \lim_{\xi \searrow 0} \frac{1}{2}\left(u\left(\vec{x} + \xi \vec{n}_{\mathfrak{I},\Gamma}\right) + u\left(\vec{x} - \xi \vec{n}_{\mathfrak{I},\Gamma}\right)\right). \tag{9}$$

Note that the actual sign of the jump operator depends on the direction of the normal. Because there is no natural choice for the direction, that is, the sign of $\vec{n}_{\mathfrak{I},\Gamma}$ on $\Gamma_{\text{int}} \cup \mathfrak{I}$, the definition of the jump operator seems quite arbitrary. However, on every occasion at which the jump operator is used, it appears as part of a product of type $[\![-]\!]\vec{n}_{\mathfrak{I},\Gamma}$ or $[\![-]\!][\![-]\!]$, so that the result does not depend on the actual choice of the sign of $\vec{n}_{\mathfrak{I},\Gamma}$. On the boundary $\partial\Omega$, jump and average are redefined as

$$[\![u]\!](\vec{x}) := \lim_{\xi \searrow 0} u\left(\vec{x} - \xi \vec{n}_{\Gamma}\right) \text{ and } \{\!\{u\}\!\}(\vec{x}) := \lim_{\xi \searrow 0} u\left(\vec{x} - \xi \vec{n}_{\Gamma}\right);$$

In addition, one defines the inner and outer value of u as

$$u^{\text{in}}(\vec{x}) := \lim_{\xi \searrow 0} u\left(\vec{x} - \xi \vec{n}_{\Gamma}\right) \text{ for } \vec{x} \in \Gamma$$
$$u^{\text{out}}(\vec{x}) := \lim_{\xi \searrow 0} u\left(\vec{x} + \xi \vec{n}_{\Gamma}\right) \text{ for } \vec{x} \in \Gamma_{\text{int}}$$

- the broken polynomial space of total degree k:

$$\mathbb{P}_k(\mathfrak{K}_h) := \left\{ f \in L^2(\Omega); \forall\ K \in \mathfrak{K}_h : f|_K \text{ is polynomial and } \deg\left(f|_K\right) \leqslant k \right\}; \tag{10}$$

- the broken gradient ∇_h: for $u \in \mathcal{C}^1(\Omega \setminus \Gamma \setminus \mathfrak{I})$, $\nabla_h u$ denotes the gradient on the domain $\Omega \setminus \Gamma \setminus \mathfrak{I}$; in analog fashion, the broken divergence $\text{div}_h\left(\vec{u}\right)$;
- the measure $|X|$ of some set X is given as $|X| = \int_X 1 \text{dV}$;
- the species-volume-fraction $\text{fr}_{\mathfrak{s}}(K)$ for some cell $K \in \mathfrak{K}_h$: $\text{fr}_{\mathfrak{s}}(K) := |K \cap \mathfrak{s}|/|K|$, for $\mathfrak{s} \in \{\mathfrak{A}, \mathfrak{B}\}$;
- the standard-basis vector $\vec{e}_d$, for $d \in \{1, 2\}$: $\vec{e}_1 = (1, 0)$, $\vec{e}_2 = (0, 1)$;

3.2. The extended discontinuous Galerkin discretization

In order to represent the phases $\mathfrak{A}$ and $\mathfrak{B}$ as well as the interface $\mathfrak{I}$ numerically, we employ a level set formulation. We choose some sufficiently smooth scalar field $\varphi \in \mathbb{P}_r(\mathfrak{K}_h) \cap \mathcal{C}^n(\Omega)$, such that

$$\mathfrak{I} = \left\{ \vec{x} \in \Omega; \varphi(\vec{x}) = 0 \right\},$$
$$\mathfrak{A} = \left\{ \vec{x} \in \Omega; \varphi(\vec{x}) < 0 \right\} \text{ and}$$
$$\mathfrak{B} = \left\{ \vec{x} \in \Omega; \varphi(\vec{x}) > 0 \right\}.$$

For a comprehensive overview on level set methods, we refer, for example, to the textbooks of [29] or [30]. Because we consider only steady-state situations without any interface movement, level set evolution algorithms are not relevant for this work and are not being discussed. Obviously, the interface normal can be obtained from the gradient of φ, and the curvature can be calculated by Bonnet's formula:

$$\vec{n}_{\mathfrak{I}} = \frac{\nabla\varphi}{|\nabla\varphi|} \text{ and } \kappa = \text{div}\left(\frac{\nabla\varphi}{|\nabla\varphi|}\right). \tag{11}$$

The extended DG space, or XDG space, is defined as

$$\mathbb{P}_k^X(\varphi, \mathfrak{K}_h) := \left\{ f \in L^2(\Omega);\ \forall\ K \in \mathfrak{K}_h : f|_{K\cap\mathfrak{A}}, f|_{K\cap\mathfrak{B}} \text{ are polynomial, } \deg\left(f|_{K\cap\mathfrak{A}}\right) \leqslant k \text{ and } \deg\left(f|_{K\cap\mathfrak{B}}\right) \leqslant k \right\}. \tag{12}$$

We note that there exists a grid ${\mathfrak{K}_h}^{X,\varphi}$ that consists of all cut cells so that

$$\mathbb{P}_k^X(\varphi, \mathfrak{K}_h) = \mathbb{P}_k\left({\mathfrak{K}_h}^{X,\varphi}\right). \tag{13}$$

DOI: 10.1002/nme

3.3. The spatial discretization of the two-phase problem

In order to comply with the Ladyženskaja–Babuška–Brezzi condition (e.g., [31, 32]) in the case of equal physical parameters (i.e., $\rho_{\mathfrak{A}} = \rho_{\mathfrak{B}}$, $\mu_{\mathfrak{A}} = \mu_{\mathfrak{B}}$), we discretize velocity and pressure in XDG spaces of order k and $k' := k - 1$, respectively, that is,

$$(\vec{u}, p) \in \mathbb{P}_k^X(\varphi, \mathfrak{K}_h)^2 \times \mathbb{P}_{k-1}^X(\varphi, \mathfrak{K}_h) =: V_k. \tag{14}$$

To the best of our knowledge, there is no rigorous proof for the inf-sup stability of this velocity/pressure pair that would directly apply to the XDG setting. Stabilization by additional jump terms on the pressure (e.g., [17]) in the two-phase setting would lead to additional dependencies on $\nabla\vec{u}$ in the continuity equation. In [33], an inf-sup condition for this velocity/pressure pair for $1 \leqslant k \leqslant 3$ was shown for grids with hanging nodes and triangular elements. Furthermore, it is also required to introduce additional penalty terms in the bilinear form $b(-,-)$ (Equation (20)) on the interface between subdomains to derive the stability result. There is, however, some experimental evidence for the stability of this velocity/pressure pair (e.g., [34]). Also in this work, we perform extensive testing for stability (Section 5). Using the agglomeration procedure presented later in this work typically leads to a better shape regularity of the cut cells and improves the condition number of the system.

We propose the following discretization of the multiphase Navier–Stokes problem (3, 4) with jump conditions (5, 6) and boundary conditions (7) in the extended DG space: find $(\vec{u}, p) \in V_k$, such that for all $(\vec{v}, \tau) \in V_k$

$$Ns(\vec{u}, (\vec{u}, p), (\vec{v}, \tau)) = \underbrace{q(\vec{v}) + s(\vec{v}) + r(\tau)}_{=:rhsNs((\vec{v},\tau))}, \tag{15}$$

where the Navier–Stokes form $Ns(-,-,-)$ is given as the sum

$$Ns(\vec{u}, (\vec{u}, p), (\vec{v}, \tau)) = L\left((\vec{u}, p), (\vec{v}, \tau)\right) + t\left(\vec{u}, \vec{u}, \vec{v}\right). \tag{16}$$

While the trilinear form $t(-,-,-)$ represents convection terms, the bilinear Stokes form $L(-,-)$ could be further decomposed as

$$L((\vec{u}, p), (\vec{v}, \tau)) = b(p, \vec{v}) - a(\vec{u}, \vec{v}) - b(\tau, \vec{u}). \tag{17}$$

The bilinear form $b(-,-)$ represents pressure gradient and velocity divergence, and the bilinear form $a(-,-)$ represents viscous terms. Furthermore, in Equation (15) surface tension, boundary conditions for the Stokes part and body forces are represented by the linear form $s(-)$, while boundary conditions for the convective part can be found in $q(-)$ and boundary conditions for the continuity equation in $r(-)$. In addition, the multiphase Stokes problem, obtained by neglecting the convective part in Equations (3) and (4) resp. Equation (15), is discretized as

$$L((\vec{u}, p), (\vec{v}, \tau)) = s(\vec{v}) + r(\tau), \tag{18}$$

for all $(\vec{v}, \tau) \in V_k$.

The convective terms are discretized by a local Lax–Friedrichs flux,

$$t(\vec{w}, \vec{u}, \vec{v}) = -\int_\Omega \rho(\vec{u} \otimes \vec{w}) : \nabla_h \vec{v} \mathrm{dV} - \oint_{\Gamma_{\mathrm{int}} \cup \Gamma_{\mathrm{N}} \cup \mathfrak{I}} \left(\left\{\left\{\vec{u} \otimes \vec{w}\right\}\right\} \vec{n}_{\mathfrak{I},\Gamma} + (\lambda/2)[\![\vec{u}]\!]\right) \cdot [\![\rho\vec{v}]\!] \mathrm{dS}, \tag{19}$$

as used in the work of Shahbazi *et al.* [35]. Details on the stabilization parameter λ are given in Section 3.3.1. There are obviously alternatives to the Lax–Friedrichs flux, for example, one may employ an upwind flux and obtain similar results.

DOI: 10.1002/nme

Pressure gradient and velocity divergence are discretized as

$$b(p,\vec{v}) = -\int_\Omega p\ \mathrm{div}_h\left(\vec{v}\right)\mathrm{dV} - \oint_{(\Gamma\setminus\Gamma_\mathrm{N})\cup\mathfrak{I}} [\![\vec{v}]\!]\cdot\vec{n}_{\mathfrak{I},\Gamma}\ \{\{p\}\}\,\mathrm{dS}. \tag{20}$$

To discretize the viscous terms, we employ an almost standard symmetric interior penalty (SIP) method, first introduced by Arnold [36] (for an extensive analysis, see, e.g., [37]):

$$\begin{aligned} a(\vec{u},\vec{v}) = &-\int_\Omega \mu(\nabla_h\vec{u} : \nabla_h\vec{v} + (\nabla_h\vec{u})^T : \nabla_h\vec{v})\mathrm{dV} \\ &+\oint_{(\Gamma\setminus\Gamma_\mathrm{N})\cup\mathfrak{I}} \left(\left\{\left\{\mu\left(\nabla_h\vec{u}+\nabla_h\vec{u}^T\right)\right\}\right\}\vec{n}_{\mathfrak{I},\Gamma}\right)\cdot[\![\vec{v}]\!] + \left(\left\{\left\{\mu\left(\nabla_h\vec{v}+\nabla_h\vec{v}^T\right)\right\}\right\}\vec{n}_{\mathfrak{I},\Gamma}\right)\cdot[\![\vec{u}]\!]\mathrm{dS} \\ &-\oint_{(\Gamma\setminus\Gamma_\mathrm{N})\cup\mathfrak{I}} \eta[\![\vec{u}]\!]\cdot[\![\vec{v}]\!]\mathrm{dS}. \end{aligned} \tag{21}$$

This form is basically an extension of the classical SIP form for the discretization of a Laplace operator of the form $\mathrm{A}u := \mathrm{div}\left(\mu\nabla u\right)$ to the operator $\mathrm{A}\vec{u} := \mathrm{div}\left(\mu\left(\nabla\vec{u}+\nabla\vec{u}^T\right)\right)$. Because $\nabla_h\vec{u}^T : \nabla_h\vec{v} = \nabla_h\vec{v}^T : \nabla_h\vec{u}$, the bilinear form $a(\vec{u},\vec{v})$ is symmetric in $\vec{u}$ and $\vec{v}$. Details on the choice of the penalty parameter η are given in Section 3.3.2.

A disadvantage of this choice for the discretization of viscous terms is the coupling of all velocity components in the bulk phase where they could be uncoupled by exploiting the relation $\mathrm{div}\left(\mu\nabla\vec{u}^T\right) = 0$. Such a form has been proposed, for example, by Bastian and Engwer [6], where the coupling of the velocity components only occurs for those DOF that are related to cut cells, resp. the momentum jump condition Equation (6). Therefore, these discretizations result in less non-zero entries in the operator matrix than the choice proposed in this work. On the other hand, the discretization of [6] is not symmetric; to the best of our knowledge, there exists no discretization that fulfills both, symmetry and decoupling of velocity components in the bulk phase. For this work, we prioritize symmetry over sparsity.

Finally, we specify the sources of the Stokes problem,

$$\begin{aligned} s(\vec{v}) = &-\int_\Omega \vec{F}\cdot\vec{v}\mathrm{dV} + \oint_{\mathfrak{I}} \kappa\ \sigma\ (\vec{n}_{\mathfrak{I}}\cdot[\![\vec{v}]\!])\mathrm{dS} \\ &-\oint_{\Gamma_\mathrm{D}} \vec{u}_\mathrm{D}\cdot\left(\nabla_h\vec{v}\vec{n}_{\partial\Omega} + \nabla_h\vec{v}^T\vec{n}_{\partial\Omega} - \eta\vec{v}\right)\mathrm{dS}, \end{aligned} \tag{22}$$

where the first, second, and third terms represent volume force, force induced by surface tension, and Dirichlet boundary conditions for the viscous part, respectively. The Dirichlet boundary conditions for the continuity equation are given by

$$r(\tau) = \oint_{\Gamma_\mathrm{D}} \tau\ \vec{u}_\mathrm{D}\cdot\vec{n}_{\partial\Omega}\mathrm{dS}, \tag{23}$$

and for the convective by

$$q(\vec{v}) = -\oint_{\Gamma_\mathrm{D}} \left(\vec{u}_\mathrm{D}\otimes\vec{u}_\mathrm{D}\right)\vec{n}_{\mathfrak{I}}\cdot[\![\rho\vec{v}]\!]\mathrm{dS}. \tag{24}$$

3.3.1. Choice of the Local–Lax–Friedrichs parameter λ. Like in [35], the stabilization parameter for the convective terms is chosen as

$$\lambda := \max\left\{|\varrho|;\varrho\in\mathrm{spec}\left(\vec{Q}\left(\overline{\vec{u}^{\mathrm{in}}}\right)\right)\cup\mathrm{spec}\left(\vec{Q}\left(\overline{\vec{u}^{\mathrm{out}}}\right)\right)\right\},$$

where $\mathrm{spec}(\vec{Q})$ denotes the spectrum of the matrix $\vec{Q}$; the matrix $\vec{Q}(\vec{u})$ is given as

DOI: 10.1002/nme

$$\vec{Q}(\vec{u}) = \nabla_{\vec{u}} \left((\vec{u} \otimes \vec{u}) \, \vec{n}_{\mathfrak{I},\Gamma} \right) = \begin{bmatrix} 2u_1 n_1 + u_2 n_2 & u_1 n_2 \\ u_2 n_1 & u_1 n_1 + 2u_2 n_2 \end{bmatrix},$$

where $\overline{\vec{u}^{\text{in}}}$ and $\overline{\vec{u}^{\text{out}}}$ denote the mean values of $\vec{u}^{\text{in}}$ and $\vec{u}^{\text{out}}$ within two cells adjacent to the corresponding edge $E \subset \Gamma$.

3.3.2. Choice of the symmetric interior penalty parameter η. Regarding the classical SIP-method, it is well known (e.g., [36]) that the penalty parameter η must be chosen large enough to ensure coercivity of the form $a(-,-)$; that is, one has to ensure

$$a(\vec{u}, \vec{u}) \geqslant \text{const} \|\vec{u}\|_*^2 \quad \forall \vec{u} \in \mathbb{P}_k^X(\varphi, \mathfrak{K}_h)^2$$

for some suitable norm $\| - \|_*$. On the other hand, the penalty parameter should not be chosen arbitrarily large because over-penalization increases the condition number and the approximation error. Thus, it is important to choose η as small as possible, yet large enough.

The foundation of coercivity estimates for the SIP method are inverse trace inequalities (e.g., [17, 38]), which bound norms on the cell boundary by norms on the cell. These inequalities in general read as

$$\|u\|_{L^2(\partial K)}^2 \leqslant \underbrace{\frac{C(k)}{h}}_{=:\tilde{\eta}} \|u\|_{L^2(K)}^2 \quad \forall u \in \mathbb{P}_k(\{K\}),$$

where the constant $C(k)$ is in $\mathcal{O}(k^2)$.

The explicit expression for $C(k)$ and the choice of the geometric factor $1/h$ depend on the element type; sharp estimates for simplical elements were proven by Warburton and Hesthaven [39] and for tensor product elements by Burman and Ern [40]; an overview can be found in [41]. Recently, Antonietti *et al.* [42] and Cangiani *et al.* [43] developed estimates for η on polygonal and polyhedral meshes.

In the context of XDG, one has to deal with cut cells $K^X = K \cap \mathfrak{s}$, $\mathfrak{s} \in \{\mathfrak{A}, \mathfrak{B}\}$, for a background cell K. Unfortunately, we are not aware of any inverse trace inequality for cut cells K^X of general shape. However, if one assumes that $\mathfrak{I}$ is rather straight and no background cell K is cut twice (i.e., the local radius of $\mathfrak{I}$ is large in comparison with the size of K, and $\mathfrak{I} \cap K$ is simply connected) in 2D, all possible cut cells K^X fall into one of the following cases:

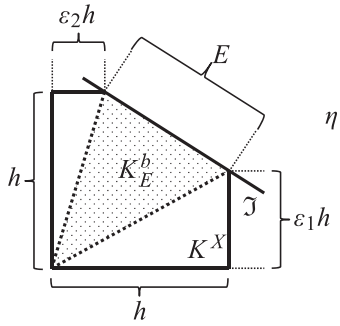
- Triangular cells: the geometric factor is $|\partial K^X|/|K^X|$ [39].
- Qadrangular cells: Estimates are proven for geometric factors $1/|E|$ as well as $|\partial K^X|/|K^X|$ [40]. For quadrilaterals with a rather short edge E, that is, if the quadrilateral approaches triangular shape, the latter choice is suited better, because $1/|E|$ may become arbitrarily large. For highly anisotropic rectangles of size $h_1 \times h_2$, $h_1 \ll h_2$, both estimate scale as $1/h_1$ in the worst case. However, these cases are avoided by the cell agglomeration procedure (Section 5.1).
- Pentagonal cells: in [43], for some face E of a polygonal cell, the inverse estimate

$$\|u\|_{L^2(E)}^2 \leqslant \underbrace{\frac{Ck^2|E|}{|K_E^b|}}_{=:\tilde{\eta}} \|u\|_{L^2(K^X)}^2,$$

  where $K_E^b \subset K^X$ is the largest possible triangle that shares edge E (Figure 1), is proven. In the cases we are interested in, as illustrated in Figure 1, it is possible to bound the geometric factor $|E|/|K_E^b|$ by $\eta_0 |\partial K^X|/|K^X|$ with some suitable $\eta_0 > 0$.

For these reasons, $|\partial K^X|/|K^X|$ is used as a geometric factor; $|\partial K^X|$ and $|K^X|$ are available from the construction of cut-cell quadrature rules (Section 4), so no significant further computational work has to be carried out. As already indicated, the main tool that we use to address the issue of small cut cells is cell agglomeration, in detail discussed in Section 5.1. Basically, if the volume fraction of some cut cell K^X is smaller than a certain threshold ($\text{fr}_{\mathfrak{s}}(K) < \alpha$), it is joined with its largest neighbor cell L in the same species $\mathfrak{s}$.

DOI: 10.1002/nme



$$\eta = \text{const}\, k^2 \frac{|E|}{|K_E^b|} = \text{const}\, k^2 \frac{1}{h} \underbrace{\frac{\sqrt{(1-\varepsilon_1)^2 + (1-\varepsilon_2)^2}}{1+\varepsilon_1\varepsilon_2}}_{\leq 2\sqrt{2}} \leq \eta_0 \frac{|\partial K^X|}{|K^X|}$$

Figure 1. Penalty estimation according to Antonietti *et al.* [42] and Cangiani *et al.* [43] for an edge $E \subset \mathfrak{I}$ of the pentagonal cut cell $K^X = K \cap \mathfrak{s}$.

The local penalty factor is set as

$$\tilde{\eta} = \eta_0 k^2 \frac{|\partial K^X|}{|K^X|} \quad \text{for each agglomerated cut cell } K^X \in \mathfrak{K}_h{}^{X,\text{agg},\varphi,\alpha}, \tag{25}$$

where the safety factor η_0 is chosen to be 4.0 in all examples demonstrated within this work. Because $\tilde{\eta}$ is piecewise constant within each cut cell, one can define the penalty parameter as

$$\begin{aligned} &\eta := \max\left\{\mu^{\text{in}}, \mu^{\text{out}}\right\} \cdot \max\left\{\tilde{\eta}^{\text{in}}, \tilde{\eta}^{\text{out}}\right\} \text{ on } \Gamma_{\text{int}} \text{ and} \\ &\eta := \mu^{\text{in}} \cdot \tilde{\eta}^{\text{in}} \text{on } \partial\Omega. \end{aligned} \tag{26}$$

Numerical experiments (Section 5.4) indicate that this choice for the penalty parameter in combination with cell agglomeration is suitable to ensure coercivity/positive definiteness of form $a(-,-)$ and that the condition number of the saddle point problem is rather independent of the position of the interface $\mathfrak{I}$.

Alternatively, one may use the ghost penalty approach introduced by Burman [15] to avoid a scaling of η that is determined by potentially infinitely small cut cells. This approach was also used by Hansbo *et al.* [14]. In the context of DG methods, it was analyzed by Massjung [24]. The drawback of the ghost penalty method is that it requires higher-order penalty terms for higher-order approximation spaces. Approximation spaces of degree 1 require penalty terms for gradients, that is, terms like $\oint_{(\Gamma\setminus\Gamma_N)\cup\mathfrak{I}} \eta[\![\nabla_h \vec{u}]\!] : [\![\nabla_h \vec{v}]\!] \text{dS}$. Higher degree approximations require penalization of higher derivatives, that is, penalty terms like $\sum_{|\beta|\leq k} \left[\!\left[\partial_h^\beta \vec{u}\right]\!\right] \cdot \left[\!\left[\partial_h^\beta \vec{v}\right]\!\right]$, where β denotes a multi-index, and k is the polynomial order of the approximation. The implementation thus becomes more and more complex with higher k. Instead, we prefer to overcome conditioning issues caused by small cut cells by a cell agglomeration procedure introduced in Section 5.

4. NUMERICAL INTEGRATION IN CUT CELLS

One actual challenge for implementing XDG methods is the numerical integration on cut cells, that is, the numerical evaluation of the integrals

$$\begin{aligned} \text{for cut cell boundaries, that is, } &\oint_{\partial K \cap \mathfrak{A}} f \,\text{dS}, \oint_{\partial K \cap \mathfrak{B}} f \,\text{dS}, \\ \text{for the surface, that is, } &\oint_{K \cap \mathfrak{I}} f \,\text{dS} \text{ and} \\ \text{for cut cell volumes, that is, } &\int_{K \cap \mathfrak{A}} f \,\text{dV}, \int_{K \cap \mathfrak{B}} f \,\text{dV}, \end{aligned}$$

DOI: 10.1002/nme

for some sufficiently smooth $f \in \mathcal{C}^n(K)$, on a reference cell $K \subset \mathbb{R}^2$, in order to evaluate the forms defined in the previous section. We address this problem by the so-called HMF procedure, proposed by Kummer and Oberlack [5], with some modifications that will be outlined later. The starting point for the construction of HMF rules is the Gauss theorem, that is,

$$\int_{K\cap\mathfrak{A}} \operatorname{div}\left(\vec{f}\right)\mathrm{dV} - \oint_{K\cap\mathfrak{I}} \vec{f}\cdot\vec{n}_{\mathfrak{I}}\mathrm{dS} = \oint_{\partial K\cap\mathfrak{A}} \vec{f}\cdot\vec{n}_{\partial K}\mathrm{dS} \tag{27}$$

for a suitable vector field $\vec{f}$. Using this, one can give a linear ansatz for unknown quadrature weights for a given set of quadrature nodes. In order to determine the right-hand side of Equation (27), we follow the original approach and construct the quadrature rule $\left(\underline{\vec{x}}^{\partial}, \underline{w}^{\partial}\right)$ by finding the discrete zero-set of the level set field φ on the one-dimensional set ∂K. For details on this and on the selection of quadrature nodes $\underline{\vec{x}}$, see [5]. The original HMF procedure, as given by definition 3, fulfills the Gauss theorem (27) only for a certain sub-space of all vector functions $\vec{f}$ in $\mathbb{P}_k(\{K\})^2$. This issue can be fixed by the modified HMF version introduced in Definition 4. Obviously, the Gauss theorem is not the only integral theorem that gives a relation between boundary, surface , and volume integrals. The precision of the HMF procedure can be further increased by also incorporating Stokes theorem, where we use the special form

$$\oint_{\mathfrak{I}\cap K} \kappa\vec{n}_{\mathfrak{I}}\cdot\vec{f} - \left(I - \vec{n}_{\mathfrak{I}}\otimes\vec{n}_{\mathfrak{I}}\right) : \nabla\vec{f}\,\mathrm{dS} = -\int_{\partial(\mathfrak{I}\cap K)} \vec{s}\cdot\vec{f}\,\mathrm{d}\ell, \tag{28}$$

where $\int_X \ldots \mathrm{d}\ell$ denotes, in the general 2D setting of this work, a zero-dimensional point measure over all points in X, κ is the curvature as defined in Equation (11), and $\vec{s}$ is tangent to $\mathfrak{I}$, pointing outward of K. The integration domains and vector fields are also illustrated in Figure 2. In 3D, $\int_X \ldots \mathrm{d}\ell$ would be a one-dimensional line integral. A third variant of HMF, which also includes this relation, is given in Definition 4.

The remainder of Section 4 is organized in the following way: Within Section 4.1, the three different variants of the HMF procedure are defined. In Section 4.2, an exemplary convergence result is presented; because HMF has been extensively tested in the original publication, we only provide a single test case to confirm these results.

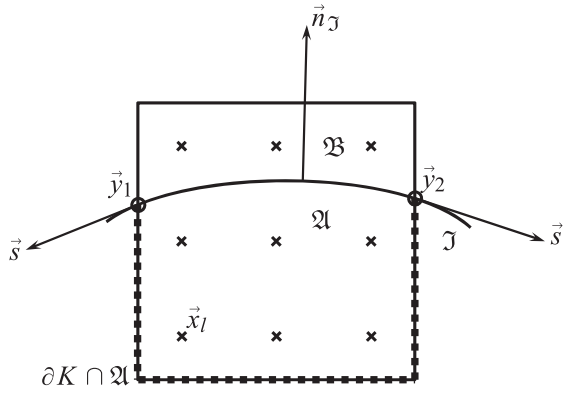


Figure 2. Illustration of the hierarchical-moment-fitting quadrature: the cell K is partitioned into the cut cells $K\cap\mathfrak{A}$ and $K\cap\mathfrak{B}$, separated by the interface $\mathfrak{I}$. Therefore, the boundary of, for example, $K\cap\mathfrak{A}$ consists of the interface part $\mathfrak{I}\cap K$, and the K-boundary part $\partial K\cap\mathfrak{A}$ (marked by dots). In 2D, for an interface that crosses the cell boundary exactly twice, the zero-dimensional point measure is just $\int_{\partial(\mathfrak{I}\cap K)} g\mathrm{d}\ell = g(\vec{y}_1) + g(\vec{y}_2)$. $\vec{s}$ denotes the outer tangent field to $\mathfrak{I}$ on $\partial(\mathfrak{I}\cap K)$. Note that the quadrature nodes $\vec{x}_l$ for all other measures are independent of the interface position.

DOI: 10.1002/nme

4.1. Three variants of hierarchical-moment-fitting rules

Definition 2 (notation of quadrature rules)
For the quadrature (numerical integration or cubature), using

$$\text{nodes } \underline{\vec{x}} = \left(\vec{x}_1, \ldots, \vec{x}_L\right) \text{ and weights } \underline{w} = (w_1, \ldots, w_L)\,,$$

of a function $f(\vec{x})$, we define the notation

$$\int_{(\underline{\vec{x}},\underline{w})}^{num} f := \sum_{l=1}^{L} w_l\, f\left(\vec{x}_l\right). \tag{29}$$

The original HMF procedure [5] is defined in the following way:

Definition 3 (original HMF procedure, two-step HMF, of order k: $HMF_A(k)$)
For a given list of quadrature nodes $\underline{\vec{x}}$, weights $\underline{w}^{\mathfrak{I}}$ and $\underline{w}^{\mathfrak{A}}$ for quadrature over surface $\mathfrak{I} \cap K$ and volume $\mathfrak{A} \cap K$, respectively, are given as the least squares solution of the following linear systems:

(1) Step 1, construction of the surface rule. Solve

$$-\oint_{(\underline{\vec{x}},\underline{w}^{\mathfrak{I}})}^{num} \vec{f} \cdot \vec{n}_{\mathfrak{I}} = \oint_{\partial K \cap \mathfrak{A}} \vec{f} \cdot \vec{n}_{\partial K} \mathrm{dS} \qquad \forall\, \vec{f} \in \mathbb{P}_k(\{K\})^2, \mathrm{div}\left(\vec{f}\right) = 0 \tag{30}$$

for surface weights $\underline{w}^{\mathfrak{I}}$ in a least square sense.

(2) Step 2, construction of the volume rule. Using the already constructed surface rule, Solve

$$\begin{aligned} \int_{(\underline{\vec{x}},\underline{w}^{\mathfrak{A}})}^{num} \mathrm{div}\left(\vec{f}\right) = \oint_{\partial K \cap \mathfrak{A}} \vec{f} \cdot \vec{n}_{\partial K} \mathrm{dS} + \oint_{(\underline{\vec{x}},\underline{w}^{\mathfrak{I}})}^{num} \vec{f} \cdot \vec{n}_{\mathfrak{I}} \\ \forall \phi \in \mathbb{P}_k(\{K\}),\, \vec{f} = \frac{1}{2}\left(\int \phi \mathrm{d}x, \int \phi \mathrm{d}y\right) \end{aligned} \tag{31}$$

for volume weights $\underline{w}^{\mathfrak{A}}$ in a least square sense.

Relations (30) and (31) represent linear systems in the quadrature weights $\underline{w}^{\mathfrak{I}}$ and $\underline{w}^{\mathfrak{A}}$. That is, for a specific $\vec{f}$ in the test space defined for relation (30), the equation related to $\vec{f}$ reads as

$$\left[\vec{f}(\vec{x}_1) \cdot \vec{n}_{\mathfrak{I}}, \ldots \vec{f}(\vec{x}_L) \cdot \vec{n}_{\mathfrak{I}}\right] \cdot \underline{w}^{\mathfrak{I}} = \oint_{\partial K \cap \mathfrak{A}} \vec{f} \cdot \vec{n}_{\partial K} \mathrm{dS}.$$

This linear relation can obviously be assembled for each member of a complete basis of the respective test space in order to obtain a linear system $M\underline{w} = b$. We choose the number of nodes and weights, L to be about 1.6 times the dimension of the test space, so that the solution of the linear system is not unique and can be chosen from an affine manifold, that is, $\underline{w} \in \underline{w}_0 + H$, $H \leqslant_{\mathbb{R}} \mathbb{R}^L$. From this solution space, the solution that has a minimal l^2 - norm is selected. This can be achieved by the `LAPACK`-function `DGELSY` in a quite efficient way.

After this review of the original procedure, the modified versions used for this work shall be given:

Definition 4 (Gauss-preserving one-step HMF of order k: $HMF_B(k)$)
For a given list of quadrature nodes $\underline{\vec{x}}$, weights $\underline{w}^{\mathfrak{I}}$ and $\underline{w}^{\mathfrak{A}}$ for quadrature over surface $\mathfrak{I} \cap K$ and volume $\mathfrak{A} \cap K$, respectively, are given as the least-squares-solution of the following linear system:

$$\int_{(\underline{\vec{x}},\underline{w}^{\mathfrak{A}})}^{num} \mathrm{div}\left(\vec{f}\right) - \oint_{(\underline{\vec{x}},\underline{w}^{\mathfrak{I}})}^{num} \vec{f} \cdot \vec{n}_{\mathfrak{I}} = \oint_{\partial K \cap \mathfrak{A}} \vec{f} \cdot \vec{n}_{\partial K} \mathrm{dS} \qquad \forall\, \vec{f} \in \mathbb{P}_k(\{K\})^2 \tag{32}$$

Finally, one can also opt to include relations from Stokes theorem into the construction process:

DOI: 10.1002/nme

Definition 5 (Gauss-preserving and Stokes-preserving one-step HMF of order k: $\mathrm{HMF}_C(k)$)
For a given list of quadrature nodes $\underline{\vec{x}}$, weights $\underline{w}^{\mathfrak{I}}$ and $\underline{w}^{\mathfrak{A}}$ for quadrature over surface $\mathfrak{I} \cap K$ and volume $\mathfrak{A} \cap K$, respectively, are given as the least-squares-solution of the following linear system:

$$\begin{cases} \int^{num}_{(\underline{\vec{x}},\underline{w}^{\mathfrak{A}})} \mathrm{div}\left(\vec{f}\right) - \oint^{num}_{(\underline{\vec{x}},\underline{w}^{\mathfrak{I}})} \vec{f}\cdot\vec{n}_{\mathfrak{I}} & = \oint_{\partial K\cap\mathfrak{A}} \vec{f}\cdot\vec{n}_{\partial K}\mathrm{dS} \\ \oint^{num}_{(\underline{\vec{x}},\underline{w}^{\mathfrak{I}})} \kappa\vec{n}_{\mathfrak{I}}\cdot\vec{f} - (I-\vec{n}_{\mathfrak{I}}\otimes\vec{n}_{\mathfrak{I}}) : \nabla\vec{f} & = -\int_{\partial(\mathfrak{I}\cap K)} \vec{s}\cdot\vec{f}\,\mathrm{d}\ell \end{cases} \quad \forall \vec{f} \in \mathbb{P}_k(\{K\})^2 \qquad (33)$$

4.2. Numerical test on hierarchical-moment-fitting-quadrature and additional remarks

An exemplary convergence behavior of the HMF quadrature is shown in Figure 3. Note that these results may indicate that there is no significant difference between $\mathrm{HMF}_\mathrm{A}(k)$ and $\mathrm{HMF}_\mathrm{B}(k)$. However, a calculation of a static droplet (Section 6.1) shows that by ensuring the numerical preservation of the Gauss theorem, one is able to reduce the error by several orders of magnitude and can further be reduced by $\mathrm{HMF}_\mathrm{C}(k)$

Finally, to obtain a quadrature rule for the domain $K \cap \mathfrak{B}$, we use the identity

$$\int_{K\cap\mathfrak{B}} f\,\mathrm{dV} = \int_K f\,\mathrm{dV} - \int_{K\cap\mathfrak{A}} f\,\mathrm{dV}.$$

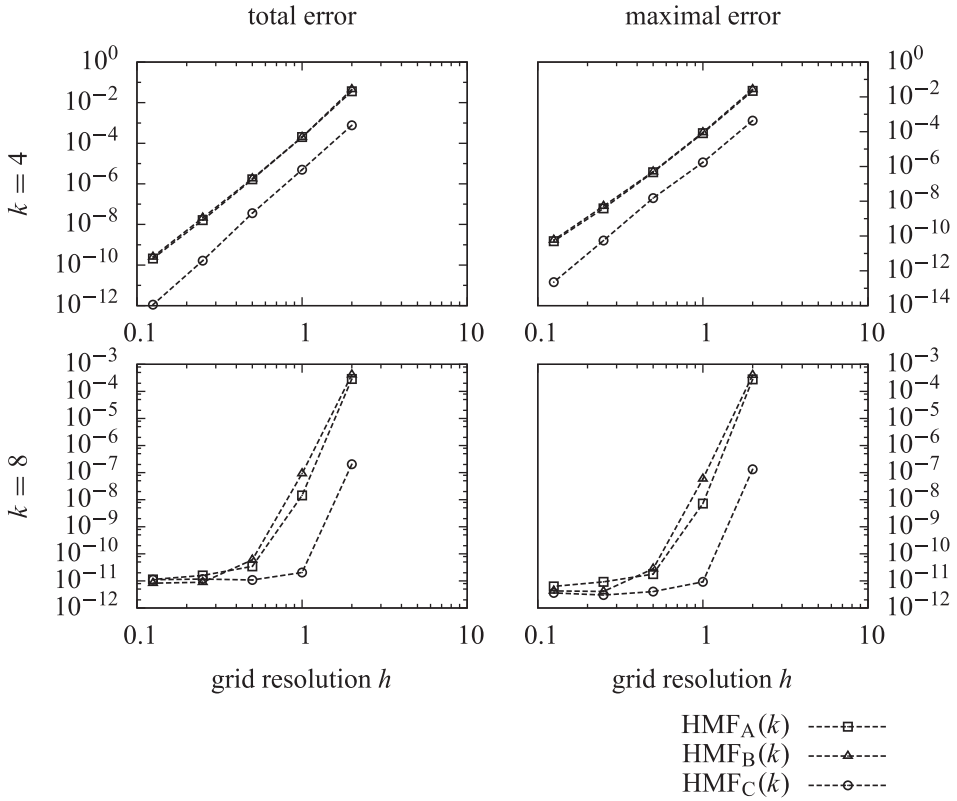


Figure 3. Convergence of hierarchical-moment-fitting quadrature, for orders $k \in \{4, 8\}$: As a test integrand, the function $f(x, y) = \cos(2x) + \sin(2y)$ is used. The domain $\Omega = (-3, 3)^2$ is discretized by grids with 3×3, 6×6, 12×12, 24×24, and 48×48 equidistant cells, while the interface is a circle with radius 2.4, that is, $\varphi = 2.4^2 - x^2 - y^2$. We define the total error as $\left(\sum_{K\in\mathfrak{K}_h}\left(\oint^{num}_{(\underline{\vec{x}},\underline{w}^{\mathfrak{I}})} f - \oint_{\mathfrak{I}\cap K} f\,\mathrm{dS}\right)^2\right)^{1/2}$ and the maximal error as $\max_{K\in\mathfrak{K}_h}\left|\oint^{num}_{(\underline{\vec{x}},\underline{w}^{\mathfrak{I}})} f - \oint_{\mathfrak{I}\cap K} f\,\mathrm{dS}\right|$.

DOI: 10.1002/nme

Then, if $\left(\underline{\vec{x}}^K, \underline{w}^K\right)$ denotes a quadrature rule – of at least an order k – for domain K, a rule for $K \cap \mathfrak{B}$ is given by

$$\left(\left[\underline{\vec{x}}^K, \underline{\vec{x}}\right], \left[\underline{w}^K, -\underline{w}^{\mathfrak{A}}\right]\right),$$

where $[\underline{a}, \underline{b}]$ denotes the concatenation of the vectors $\underline{a}$ and $\underline{b}$. By this choice, we ensure that the sum of the numerical integrals over both domains $K \cap \mathfrak{A}$ and $K \cap \mathfrak{B}$ is indeed equal to the integral over K. The boundary integral over $\partial K \cap \mathfrak{B}$ can be handled analogously.

5. TAMING THE CONDITION NUMBER

In general, the linear system given by the linearized multiphase Navier–Stokes problem (15) or the multiphase Stokes problem (18) may still have a very high condition number, especially if there are some cut cells $K^X = K \cup \mathfrak{A}$ or $K^X = K \cup \mathfrak{A}$ with either a small $\mathfrak{A}$- or $\mathfrak{B}$ - fraction (i.e., $\mathrm{fr}_{\mathfrak{A}}(K) \ll 1$ or $\mathrm{fr}_{\mathfrak{B}}(K) \ll 1$).

Hierarchical-moment-fitting-quadrature rules in general have some negative weights. Therefore, it cannot be guaranteed that the mass matrix is positive definite, that is, there may be some non-zero $u \in \mathbb{P}_k^X(\mathfrak{K}_h)$ for which

$$\int_{(\underline{\vec{x}}, \underline{w})}^{num} u^2 \mathrm{dV} \leqslant 0,$$

in some cut cell with an HMF-quadrature rule with nodes and weights $\left(\underline{\vec{x}}, \underline{w}\right)$. In numerical experiments, this behavior was only observed in rather small cut cells. Obviously, this violates the basic property that the norm of any non-zero element always has to be positive; therefore, with respect to the norm induced by the HMF quadrature, the XDG space $\mathbb{P}_k^X(\mathfrak{K}_h)$ would not be a pre-Hilbert space anymore. These issues are addressed by the following three-staged preprocessing procedure:

(1) The amount of cut cells with very small species fraction is drastically reduced by *cell agglomeration* where small cut cells are joined with their largest neighbor cell. This procedure helps in controlling the condition number with respect to arbitrary positions of the level set. It also drastically reduces the amount of troubled cells with indefinite mass matrix blocks, because these usually occur in small cells, where HMF quadrature has an intrinsic tendency to produce a large amount of negative weights.
(2) Although cell agglomeration is very helpful in eliminating cells containing elements with a negative norm, it cannot be guaranteed that this issue is completely resolved. Therefore, one still has to account for this possibility. In order to resolve the remaining problems, a rather simple modification to the Cholesky decomposition allows to remove elements with negative norm from $\mathbb{P}_k^X(\mathfrak{K}_h)$, resp. from V_k.
(3) Finally, block Jacobi preconditioning is applied to further reduce the condition number.

The first and second stages actually reduce the dimension of the problem, while the third stage is an equivalence transformation that does not change the approximation space, and therefore the solution, any further. These stages can be defined, resp. implemented, in a fashion of row and column operations applied to the saddle point system. In order to precisely specify them, we recall some basic linear algebra. At first, a canonical basis of V_k is introduced:

Definition 6 (Basis of $\mathbb{P}_k(\mathfrak{K}_h)$ and V_k)
Let $(\phi_{j,n})_{j=1,\ldots,J;\ n=1,\ldots,N}$ be an orthonormal basis of $\mathbb{P}_k(\mathfrak{K}_h)$, where j denotes the cell index, that is, $\mathrm{supp}(\phi_{j,n}) = \overline{K_j}$, and n is called the mode index. Then, let $\underline{\vec{\phi}} = \left(\vec{\phi}_{(\mathfrak{s},j,d,n)}\right) = \left(\vec{\phi}_1, \ldots, \vec{\phi}_I\right)$ be a basis of V_k with the following properties and indexing conventions:

- $\mathrm{supp}\left(\vec{\phi}_{(\mathfrak{s},j,d,n)}\right) = \overline{K_j \cap \mathfrak{s}}$, that is, $\mathfrak{s}$ denotes the species, while j denotes the cell index.

DOI: 10.1002/nme

- For some fixed species $\mathfrak{s}$, the cell index j may only be a cell in which the respective species is actually present, that is, form the set $\{1 \leqslant j \leqslant J \,|\, \mathrm{fr}_{\mathfrak{s}}\left(K_j\right) > 0\}$ containing the indices of all cells that are at least partially occupied by species $\mathfrak{s}$.
- $d \in \{1, \ldots, D+1\}$ denotes the velocity components ($1 \leqslant d \leqslant D$) resp. the pressure ($d = D+1$), that is, $\vec{\phi}_{\mathfrak{s},j,d,n} = \chi_{\mathfrak{s}} \phi_{j,n} \vec{e}_d$. The symbol χ_S denotes the characteristic function for some set S. Throughout this work, usually the spatial dimension $D = 2$.
- n denotes a polynomial index and depends on d, because for the pressure space ($d = D+1$) the polynomial degree is one less than that for the velocity;

Note that the basis functions may be indexed in two different ways, either by a multi-index $\left(\text{as } \vec{\phi}_{(\mathfrak{s},j,d,n)}\right)$ that distincts between cells, equation components, and species, or by a single number index $\left(\text{as } \vec{\phi}_i\right)$. In this sense, the tuple $(\mathfrak{s}, j, n, d)$ denotes a bijection between all valid combinations of $\mathfrak{s}$, j, n, and d and the set $\{1, 2, \ldots, I\}$ with $I := \dim(V_k)$. For a better understanding, these notations are also illustrated in a 1D example (Section 5.1.1).

With respect to the basis $\underline{\vec{\phi}}$, the matrix of the linearized Navier–Stokes, resp. the Stokes problem, is given by

$$\mathcal{L}_{ij} := Ns\left(\vec{u}^0, \vec{\phi}_i, \vec{\phi}_j\right) \text{ resp. } \mathcal{L}_{ij} := L\left(\vec{\phi}_i, \vec{\phi}_j\right) \tag{34}$$

for some fixed linearization velocity $\vec{u}^0$. The matrix $\mathcal{L}$ has the classical structure of a saddle point problem, that is,

$$\mathcal{L} = \begin{bmatrix} \mathcal{L}_{\mathrm{cd}} & \mathcal{L}_{\nabla} \\ \mathcal{L}_{\mathrm{div}} & 0 \end{bmatrix}$$

with $\mathcal{L}_{\mathrm{div}} = \mathcal{L}_{\nabla}^T$. In the previously introduced multi-index notation, one identifies the individual block matrices as

$$\begin{aligned} \text{convection–diffusion:} \quad & \mathcal{L}_{\mathrm{cd}} = \mathcal{L}_{(-,-,\{1,\ldots,D\},-)(-,-,\{1,\ldots,D\},-)}, \\ \text{pressure gradient:} \quad & \mathcal{L}_{\nabla} = \mathcal{L}_{(-,-,\{1,\ldots,D\},-)(-,-,D+1,-)}, \\ \text{velocity divergence:} \quad & \mathcal{L}_{\mathrm{div}} = \mathcal{L}_{(-,-,D+1,-)(-,-,\{1,\ldots,D\},-)}. \end{aligned}$$

Note that any change of basis is equivalent to a symmetric preconditioning; let

$$\underline{\vec{\theta}} = \left(\vec{\theta}_1, \ldots \vec{\theta}_{I'}\right)$$

be a basis of some sub-space of V_k of dimension I', that is, $\underline{\vec{\theta}}\mathbb{R}^{I'} \leqslant_{\mathbb{R}} \underline{\vec{\theta}}\mathbb{R}^{I} = V_k$, linked by an $I \times I'$ – matrix A, with $I' \leqslant I$, that is,

$$\underline{\vec{\theta}} = \underline{\vec{\phi}} A, \text{ resp., } \vec{\theta}_n = \sum_m A_{mn} \vec{\phi}_m.$$

Let $\mathcal{Q}$ be the matrices associated to $L(-,-)$ with respect to $\underline{\vec{\theta}}$, that is, $\mathcal{Q}_{ij} = L\left(\vec{\theta}_i, \vec{\theta}_j\right)$. Then, a simple calculation shows that

$$\mathcal{Q}_{ij} = L\left(\sum_m A_{mi} \vec{\phi}_m, \sum_n A_{nj} \vec{\phi}_n\right) = (A^T \mathcal{L} A)_{ij}.$$

By these basic tools, the three-stage procedure mentioned previously can be written as a product

$$\mathcal{Q} = \left(E^T C^T B^T\right) \mathcal{L}(BCE), \tag{35}$$

where B, C, and E represent cell agglomeration, elimination of negative norm elements and block-Jacobi preconditioning, respectively. E is invertible, while B and C are non-quadratic.

DOI: 10.1002/nme

5.1. Stage 1: cell agglomeration

First, two lists $\mathrm{agg}^{\alpha}_{\mathfrak{A}}, \mathrm{agg}^{\alpha}_{\mathfrak{B}} \subset {\mathfrak{K}_h}^2$ of cell pairs to agglomerate are determined by Algorithm 1, for an agglomeration threshold $0 \leqslant \alpha < 1$. Therefore, one requires

Definition 7 (edge neighbors)
The edge neighbors of some cell K are those cells $L \in \mathfrak{K}_h$ that share at least an edge (not only a vertex), that is, $\oint_{K \cap L} 1 \mathrm{dS} > 0$;

Algorithm 1 Identifying cells to agglomerate

Output: sets $\mathrm{agg}^{\alpha}_{\mathfrak{A}}, \mathrm{agg}^{\alpha}_{\mathfrak{B}} \in {\mathfrak{K}_h}^2$, holding pairs of cells to agglomerate
  set $\mathrm{agg}^{\alpha}_{\mathfrak{A}} := \{\}$, $\mathrm{agg}^{\alpha}_{\mathfrak{B}} := \{\}$
  for all species $\mathfrak{s} \in \{\mathfrak{A}, \mathfrak{B}\}$ **do**
    for all cells $K \in \mathfrak{K}_h$, with $\mathrm{fr}_{\mathfrak{s}}(K) > 0$ **do**
      if $\mathrm{fr}_{\mathfrak{s}}(K) < \alpha$ **then**
        select cell $L_{\max}$ form all edge-neighbors L of K, where $\mathrm{fr}_{\mathfrak{s}}(L)$ is maximal.
        $\mathrm{agg}^{\alpha}_{\mathfrak{s}} \leftarrow \mathrm{agg}^{\alpha}_{\mathfrak{s}} \cup \{(L_{\max}, K)\}$, i.e. add the pair $(L_{\max}, K)$ to $\mathrm{agg}^{\alpha}_{\mathfrak{s}}$
      end if
    end for
  end for

Based upon these lists, an agglomerated XDG space whose elements are polynomial in all agglomerated cells can be defined.

Definition 8 (Agglomerated XDG space and grid)

$$\mathbb{P}_k^{X,\mathrm{agg},\alpha}(\varphi, \mathfrak{K}_h) := \left\{ f \in \mathbb{P}_k^X(\varphi, \mathfrak{K}_h) | \forall\, \mathfrak{s} \in \{\mathfrak{A}, \mathfrak{B}\} : \forall (K, L) \in \mathrm{agg}^{\alpha}_{\mathfrak{s}} : f|_{(K \cup L) \cap \mathfrak{s}} \text{ is polynomial} \right\}. \tag{36}$$

In addition, the agglomerated grid ${\mathfrak{K}_h}^{X,\mathrm{agg},\varphi,\alpha}$ is defined by the property

$$\mathbb{P}_k^{X,\mathrm{agg},\alpha}(\varphi, \mathfrak{K}_h) = \mathbb{P}_k\left(\varphi, {\mathfrak{K}_h}^{X,\mathrm{agg},\varphi,\alpha}\right). \tag{37}$$

Obviously, the agglomerated XDG space is a sub-space of the original XDG space. By analogy to Equation (14), we define the agglomerated $(k, k-1)$ – velocity/pressure space

$$V_k^{\alpha} := \mathbb{P}_k^{X,\mathrm{agg},\alpha}(\varphi, \mathfrak{K}_h)^2 \times \mathbb{P}_{k-1}^{X,\mathrm{agg},\alpha}(\varphi, \mathfrak{K}_h), \tag{38}$$

which is a sub-space of V_k, that is, $V_k^{\alpha} \leqslant_{\mathbb{R}} V_k$. The matrix of the problem on the agglomerated space, $B^T \mathcal{L} B$, would be uniquely determined by the choice of a basis $\underline{\vec{\phi}}^{\mathrm{agg}}$ of V_k^{α}. However, within what remains of this section, an outline of how $B^T \mathcal{L} B$ can be obtained from $\mathcal{L}$ by a series of row (multiplication of $\mathcal{L}$ with B^T from the left) and column operations (multiplication of $\mathcal{L}$ with B from the right) is given, because this actually reflects our implementation on which all numerical results are based.

Because V_k^{α} is a subspace of V_k, any basis of V_k^{α} can be expressed as a linear combination of the basis $\underline{\vec{\phi}}$ of V_k: let $(K_{j_1}, K_{j_2}) \in \mathrm{agg}^{\alpha}_{\mathfrak{s}}$ be an arbitrary agglomeration pair for species $\mathfrak{s}$. A basis in the agglomerated cell $K^{\mathrm{agg}}_{j_1} := K_{j_1} \cup K_{j_2}$ is given by

$$\underline{\vec{\phi}}^{\mathrm{agg}}_{(\mathfrak{s},j_1,d,-)} = \left(\underline{\vec{\phi}}_{(\mathfrak{s},j_1,d,-)}, \underline{\vec{\phi}}_{(\mathfrak{s},j_2,d,-)}\right) \begin{bmatrix} I_N \\ E_{(j_1,j_2)} \end{bmatrix}. \tag{39}$$

The matrix $E_{(j_1,j_2)}$ is uniquely determined – see also the example given later in Section 5.1.1 – by the property that the polynomials $\underline{\vec{\phi}}_{(\mathfrak{s},j_2,d,-)} E_{(j_1,j_2)}$ are the continuation of the polynomials

DOI: 10.1002/nme

$\underline{\vec{\phi}}_{(\mathfrak{s},j_1,d,-)}$ from cell K_{j_1} to K_{j_2}, or equivalently that the basis functions defined by $\underline{\vec{\phi}}^{\text{agg}}_{(\mathfrak{s},j_1,d,-)}$ and all their derivatives must be continuous. Regarding the valid cell indexing of the agglomerated basis functions $\vec{\phi}^{\text{agg}}_{(\mathfrak{s},j,d,n)}$, note that the used convention is to remove the index of the second cell of an agglomeration pair from the set of valid indices. The non-zero blocks of B are given by

$$\begin{aligned} B_{(\mathfrak{s},j,d,-)(\mathfrak{s},j,d,-)} &= I_N \\ B_{(\mathfrak{s},j_2,d,-)(\mathfrak{s},j_1,d,-)} &= E_{(j_1,j_2)} \forall (K_{j_1}, K_{j_2}) \in \text{agg}^{\alpha}_{\mathfrak{s}}. \end{aligned}$$

The product $B^T \mathcal{L} B$ is equal to the following row and column operations: For all species $\mathfrak{s}$, and all agglomeration pairs $(K_{j_1}, K_{j_2}) \in \text{agg}^{\alpha}_{\mathfrak{s}}$, ...

1. Add rows and columns:

$$\begin{aligned} \mathcal{L}_{(\mathfrak{s},j_1,d,-)(-,-,-,-)} &\leftarrow \mathcal{L}_{(\mathfrak{s},j_1,d,-)(-,-,-,-)} + E^T_{(j_1,j_2)} \mathcal{L}_{(\mathfrak{s},j_2,d,-)(-,-,-,-)}, \\ \mathcal{L}_{(-,-,-,-)(\mathfrak{s},j_1,d,-)} &\leftarrow \mathcal{L}_{(-,-,-,-)(\mathfrak{s},j_1,d,-)} + \mathcal{L}_{(-,-,-,-)(\mathfrak{s},j_2,d,-)} E_{(j_1,j_2)}, \end{aligned}$$

   for all d.
2. Delete rows/columns with index $(\mathfrak{s}, j_2, -, -)$ from $\mathcal{L}$.

Note that these operations require *mutable* sparse matrix data structures that enable a rather 'cheap' insertion or deletion, optimally as $\mathcal{O}(1)$ operations. In our actual implementation, we use a row-based linked list sparse matrix structure (LIL). This is essentially a list of lists: The index into the first list is the row index; for each row of the matrix, a list of pairs containing the column index the respective matrix entry is stored. This is convenient, that is, easy to implement, but not very efficient. For DG methods, it would obviously be much more efficient to use, for example, a LIL-block matrix structure. There, the index into the first list is the *block*-row-index, and for each block row one stores pairs containing the *block*-column-index and the respective dense block matrix. For this work, we are exclusively using the PARDISO solver library [44–46]; therefore, the matrix has to be converted into a compressed row format (CSR) when it is passed to the solver. If the matrix is copied anyway, another option would be to keep track of matrix changes in suitable dynamic data structures, for example, store the product BCE in a temporary LIL structure. A special convention may be used to omit storing diagonal ones in order to save memory. The final matrix $\mathcal{Q} = (BCE)^T \mathcal{M} (BCE)$ should be stored in the format required by the solver, in our case, CSR. $\mathcal{Q}$ can be computed by a sparse matrix-matrix multiplication algorithm that, for example, accepts LIL as an input and provides CSR as an output.

5.1.1. 1D-example. The notations and procedures introduced previously should be illustrated by the following simple example. For the sake of simplicity, only a non-extended DG method in a 1D setting, for polynomial degree $k = 1$ and three cells is considered: $K_1 = (-1, 1)$, $K_2 = (1, 3)$, $K_3 = (3, 5)$; an exemplary basis for $k = 1$ is

$$\begin{aligned} \text{in } K_1: \quad \phi_{(1,1)} &:= \quad \phi_1 := \quad 1 \cdot \chi_{K_1} \phi_{(1,2)} := \quad \phi_2 := \quad x \cdot \chi_{K_1} \\ \text{in } K_2: \quad \phi_{(2,1)} &:= \quad \phi_3 := \quad 1 \cdot \chi_{K_2} \phi_{(2,2)} := \quad \phi_4 := \quad (x-2) \cdot \chi_{K_2} \\ \text{in } K_3: \quad \phi_{(3,1)} &:= \quad \phi_5 := \quad 1 \cdot \chi_{K_3} \phi_{(3,2)} := \quad \phi_6 := \quad (x-4) \cdot \chi_{K_3} \end{aligned}$$

Here, also the two different schemes for indexing the basis $\underline{\phi}$ are illustrated: either by tuples (j, n) or by a single index. Obviously, this basis is not orthonormal with respect to the usual inner product, but for this example, orthonormality is irrelevant. Here, K_2 should be agglomerated to K_1, the cell K_3 remains unchanged. Then, an exemplary basis in the agglomerated cell $K_1^{\text{agg}} := K_1 \cup K_2$ is

$$\phi^{\text{agg}}_{(1,1)} := \phi^{\text{agg}}_1 := 1 \cdot \chi_{K_1 \cup K_2}, \ \phi^{\text{agg}}_{(1,2)} := \phi^{\text{agg}}_2 := x \cdot \chi_{K_1 \cup K_2}.$$

As a result, one can describe $\underline{\phi}^{\text{agg}}_{(1,-)}$ in terms of the original basis on cells K_1 and K_2 as

DOI: 10.1002/nme

$$\underbrace{\left(\phi^{\text{agg}}_{(1,1)}, \phi^{\text{agg}}_{(1,2)}\right)}_{=\underline{\phi}^{\text{agg}}_{(1,-)}} = \underbrace{\left(\phi_{(1,1)}, \phi_{(1,2)}, \phi_{(2,1)}, \phi_{(2,2)}\right)}_{=\left(\underline{\phi}_{(1,-)}, \underline{\phi}_{(2,-)}\right)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 2 \\ 0 & 1 \end{bmatrix}.$$

In addition, we define $K_3^{\text{agg}} := K_3$ and the corresponding basis by

$$\phi^{\text{agg}}_{(3,1)} := \phi^{\text{agg}}_3 := 1 \cdot \chi_{K_3},\ \phi^{\text{agg}}_{(3,2)} := \phi^{\text{agg}}_4 := (x-4) \cdot \chi_{K_3},$$

so the complete basis transformation reads as

$$\underline{\phi}^{\text{agg}} = \underline{\phi} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{=B}.$$

Note that in our notation, there is no cell K_2^{agg}, that is, the cell K_2 has been removed from the agglomerated cells grid, which consists of K_1^{agg} and K_3^{agg} only. For this example, let $\mathcal{L} \in \mathbb{R}^{6\times 6}$ be the matrix of some bilinear form with respect to the original basis $\underline{\phi}$. Then, the matrix with respect to the agglomerated basis $\underline{\phi}^{\text{agg}}$ is $B^T \mathcal{L} B$, which is equivalent to the following row and column operations on $\mathcal{L}$:

1. Adding the $\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}^T$ –th of rows $(2,3)$ to rows $(1,2)$.
2. Adding the $\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$ –th of columns $(2,3)$ to columns $(1,2)$.
3. Deleting rows 2 and 3 as well as columns 2 and 3 from $\mathcal{L}$.

End of Example.

5.2. *Stage 2: eliminating indefinite elements*

Definition 9 (Inverse Cholesky decomposition)
For a symmetric, positive definite matrix Q, the inverse of its Cholesky factor shall be denoted by $\text{chol}^{-1}(Q)$, that is,

$$(\text{chol}^{-1}(Q))^T Q \text{chol}^{-1}(Q) = I_N,$$

where $\text{chol}^{-1}(Q)$ is an upper triangular, invertible $N \times N$ matrix.

Let

$$\mathcal{M}_{ij} := \int_\Omega \vec{\phi}_i \cdot \vec{\phi}_j \text{dV} \tag{40}$$

be the mass matrix with respect to the basis $\underline{\vec{\phi}}$. The mass matrix with respect to $\underline{\vec{\phi}}^{\text{agg}}$ is then given by $B^T \mathcal{M} B$. The diagonal blocks of this are, in multi-index notation, $(B^T \mathcal{M} B)_{(\mathfrak{s},j,d,-)(\mathfrak{s},j,d,-)} =: Q$. A transformation by the inverse Cholesky factor of the mass matrix is equivalent to an orthonormalisation of the XDG basis, because the mass matrix is transformed to the identity matrix. For uncut cells, these blocks are definitely positive definite, at least if sufficiently accurate quadrature rules are used to compute them. For the cut cells treated with HMF quadrature, this might not be the case because of potentially negative weights, as outlined previously. In this case, however, Q contains

DOI: 10.1002/nme

some positive definite sub-matrix, on which the Cholesky factorization/orthonormalisation can be performed as usual. Ignoring rows/columns with a negative diagonal element in algorithm 2 is therefore in fact equivalent to removing XDG basis vectors that have a negative norm with respect to the HMF quadrature. Therefore, the transformation matrix C of the second stage of the preprocessing is set to

$$C_{(\mathfrak{s},j,d,-)(\mathfrak{s},j,d,-)} = \mathrm{mchol}^{-1}\left((B^T \mathcal{M} B)_{(\mathfrak{s},j,d,-)(\mathfrak{s},j,d,-)}\right),$$

where the modified Cholesky factorization $\mathrm{mchol}^{-1}(-)$ is defined as in Algorithm 2.

Algorithm 2 Modified inverse Cholesky- resp. LDL-factorization, i.e. $A = \mathrm{mchol}^{-1}(Q)$, resp. $A = \mathrm{mldl}^{-1}(Q)$. In difference to the classical inverse Cholesky/LDL, the algorithm works on indefinite/singular matrices Q, but may produce zero-rows in the output A.

Input: some symmetric matrix $Q \in \mathbb{R}^{N\times N}$.

Output: an upper-diagonal matrix $A \in \mathbb{R}^{N\times N}$, so that $A^T Q A = D$. If Q is symmetrically positive definite, D is the identity matrix and A is invertible. Otherwise, D is a diagonal matrix containing only entries 0 and 1 in the 'mchol^{-1}'-case resp. 0, -1 and $+1$ in the 'mldl^{-1}'-case.

```
set A = I_N
for k = 1 to N do
    { σ ← Q_kk    for mchol^{-1}(Q)
    { σ ← |Q_kk|  for mldl^{-1}(Q)
    if σ > 0 then                                   ▷ standard Cholesky branch
        ρ ← 1/√σ
        Q_{k-} ← ρ Q_{k-}                           ▷ scale row k by ρ
        Q_{-k} ← ρ Q_{-k}                           ▷ scale column k by ρ; now, |Q_kk| = 1
        A_{-k} ← ρ A_{-k}                           ▷ perform the same column operation on A
        for i = (k + 1) to N do
            Δ ← Q_ik sign(Q_kk)
            Q_{i-} ← Q_{i-} − Δ Q_{k-}              ▷ by row add., elim. all entries below Q_kk
            Q_{-i} ← Q_{-i} − Δ Q_{-k}              ▷ by col. add., elim. all entries right of Q_kk
            A_{-i} ← A_{-i} − Δ A_{-k}              ▷ perform the same column operation on A
        end for
    else                                            ▷ the k–th basis element contains some negative norm
        A_{-k} ← 0
    end if
end for
```

As already mentioned, cells in which the mass matrix becomes indefinite are quite rare, as long as a sufficiently large agglomeration constant α is used. The preprocessed operator matrix will have zero-rows in such rare cells. Although one option would be to delete such rows, it is from an implementation perspective easier to just reset the diagonal entry to 1, and the corresponding entry on the right-hand side to 0. This allows us to still maintain the same block size for all cells, resulting in simpler data structures and algorithms.

5.3. Stage 3: block preconditioning

After stage 2, a positive definite mass matrix – under the assumption that we ignore zero rows and columns – is guaranteed. Furthermore, the operator matrix $(C^T B^T \mathcal{L} B C)$ is transformed to an orthonormal basis in the agglomerated XDG space. With respect to the condition number of the system, this still might not be the most optimal choice. Two variants for the third stage will be considered in the numerical studies. At first, simply the identity matrix, that is,

$$E = I. \tag{41}$$

DOI: 10.1002/nme

As the second variant, the block diagonal matrix

$$\begin{aligned} E_{(-,j,d,-)(-,j,d,-)} &= \mathrm{mldl}^{-1}\left(\frac{1}{2}\left(R + R^T\right)\right) && \text{for } d \in \{1,\dots,D\}(\text{velocity}) \\ E_{(-,j,d,-)(-,j,d,-)} &= I_N && \text{for } d = D+1(\text{pressure}) \\ & && \text{where } R := (C^T B^T \mathcal{L} B C)_{(-,j,d,-)(-,j,d,-)} \end{aligned} \tag{42}$$

is used. The modified inverse LDL factorization $\mathrm{mldl}^{-1}(-)$ is defined as in Algorithm 2. Essentially, this is a change of the XDG basis in a way that the symmetric part of the block diagonals of the convection–diffusion operator become diagonal matrices that contain only 0, −1, and +1 entries.

5.4. Numerical studies on condition number

By a numerical study, it is verified that the choice for the penalty parameter η (Section 3.3.2) in combination with the preprocessing presented previously (i) ensures positive definiteness of the diffusive part

$$\mathcal{Q}_{(-,-,\{1,\dots,D\},-)(-,-,\{1,\dots,D\},-)} =: \mathcal{Q}_{\mathrm{cd}} \tag{43}$$

of the preprocessed saddle point system and (ii) that the condition number of the whole saddle point problem is still manageable, despite the presence of rather small cut cells. Only the Stokes problem is investigated; then $\mathcal{Q}_{\mathrm{cd}}$ is symmetric, and positive definiteness can be tested by performing a Cholesky factorization on $\mathcal{Q}_{\mathrm{cd}}$, at least for rather small systems.

Four basic configurations were investigated, as illustrated in Figure 4: A circular interface shifted to different positions, a circular interface varied in radius, an interface that is parallel to the grid cell orientation, and a fourth one that is diagonal. Special care was taken to include also 'difficult' level sets, which directly or nearly intersect cell corners, or which are tangential to cell edges. Details on the chosen level set functions for each configuration are also given in Figure 4. The DG polynomial order for velocity and pressure is chosen as $(k, k') = (3, 2)$, and two different viscosity pairings are investigated: equal viscosities, that is, $\mu_{\mathfrak{A}} = \mu_{\mathfrak{B}} = 1$, and viscosities for air/water, that is, $\mu_{\mathfrak{A}} = 17.1 \cdot 10^{-6}$, $\mu_{\mathfrak{B}} = 10^{-3}$. The steady Stokes problem does not depend on the density ρ. For all three configurations, results were obtained for different agglomeration thresholds

$$\alpha \in \{0, 0.005, 0.01, 0.05, 0.1, 0.2\},$$

and for the two different block preconditioning options proposed in Section 5.3, see Equations (41) and (42).

For all tests, $\mathcal{Q}_{\mathrm{cd}}$ was tested to be positive definite (by performing a Cholesky decomposition), except for certain cases with $\alpha = 0$ in which cells of measure zero occur. This also indicates that the penalization parameter η (Equation (25)) is sufficiently large for the tested cases. Furthermore, it could be observed that for an agglomeration threshold above approximately 0.1, the condition number of $\mathcal{Q}$ became rather independent (minimum and maximum within one magnitude) of the level set position, for the block preconditioning option defined in Equation (42). The condition numbers for $\mathcal{Q}$ and $\mathcal{Q}_{\mathrm{cd}}$ were estimated using the `condest`-function from GNU Octave (version 4.0.0, 64 bit cygwin).

Detailed results are given in Tables I–III and Figures 5 and 6. With respect to the condition number of the diffusion operator matrix $\mathcal{Q}_{\mathrm{cd}}$, the block preprocessing (42) performs better than (41) in all cases, usually by one or two orders of magnitude, as long as a sufficiently large agglomeration factor $\alpha \geqslant 0.01$ is used. With respect to the condition number of the full saddle point matrix $\mathcal{Q}$, it performs usually better, with some exceptions where the condition number is slightly worse than for the block preconditioning option (41). Smaller agglomeration thresholds than 0.01 are not recommended, even though block preconditioning option (41) typically performs better than (42). Based on these tests, we choose an agglomeration factor $\alpha = 0.1$, and block preconditioning (42) for all further examples in this work.

By comparing Tables I ($\mu_{\mathfrak{A}} = 17.1 \cdot 10^{-6}$, $\mu_{\mathfrak{B}} = 10^{-3}$) and II ($\mu_{\mathfrak{A}} = \mu_{\mathfrak{B}} = 1$), it becomes apparent that the viscosity ratio has a certain impact on the condition number. Not surprisingly, the

DOI: 10.1002/nme

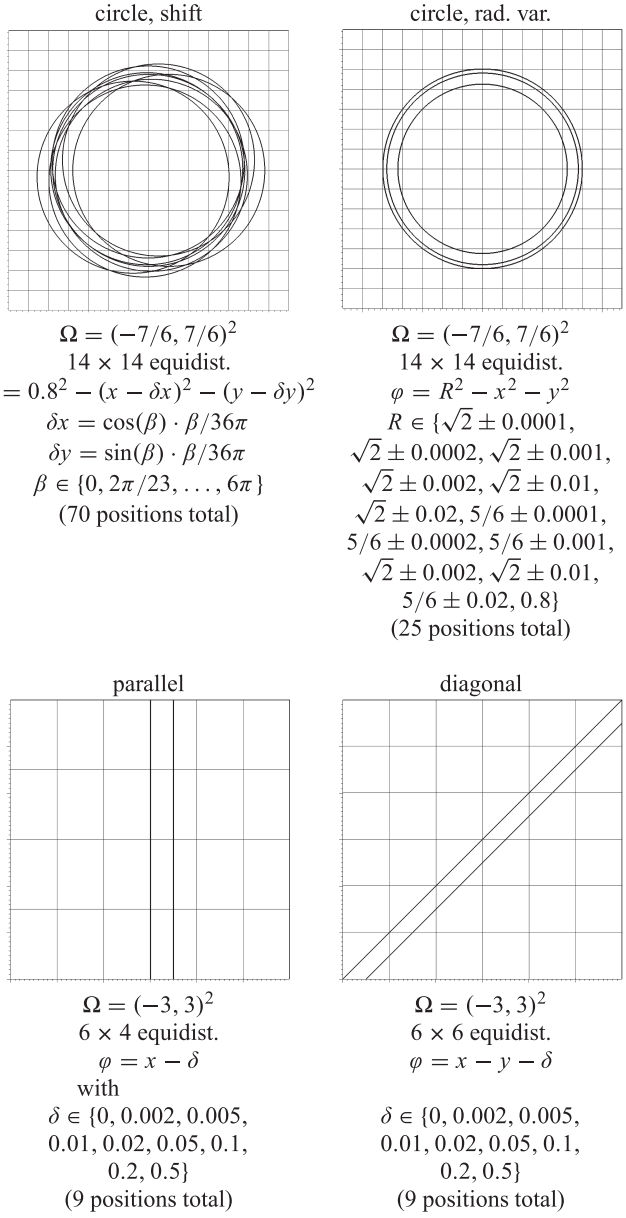


Figure 4. Grid and level set positions for the condition number tests. Note that not all level sets used for the test are shown, only exemplary ones. Most of the level sets are quite close together, so they would not be distinguishable in this plot.

condition numbers are much lower for the latter case. Especially for rather similar viscosity values, the preprocessing option (42) performs much better.

In order demonstrate the competitiveness of our method, we compare the condition number of the Stokes system $\mathcal{Q}$ against the numbers reported in the work of Hansbo *et al.* [14]. There, $\Omega = (-1, 1)$, $\mathfrak{I} = \{\vec{x}; |\vec{x}| = 1/2\}$, $\mu_{\mathfrak{A}} = \mu_{\mathfrak{B}} = 2$ and continuous P1-iso-P2 triangular elements are used; that is, for pressure, a triangular grid on 41×41 equidistant nodes is used, while the velocity is represented on a uniformly refined grid with 81×81 nodes. For both, pressure and velocity, linear ansatz functions are used. We compare this against our methods using a quadrilateral grid of 40×40 equidistant cells and polynomial orders $(2, 1)$ and $(3, 2)$ for the preprocessing option (42):

DOI: 10.1002/nme

Table I. For $\mu_{\mathfrak{A}} = 17.1 \cdot 10^{-6}$, $\mu_{\mathfrak{B}} = 10^{-3}$ the minimum and maximum condition number of the diffusion matrix $\mathcal{Q}_{\text{cd}}$ (table rows marked with D) and the full saddle point matrix $\mathcal{Q}$ (table rows marked with S) for all four different test configurations over various level set positions (Figure 4), for different agglomeration thresholds α and for two different block preconditioning options, as defined in Equations (41) and (42).

			Circle, shift test		Circle, rad. var. test		Parallel test		Diagonal test	
α			pc. (41)	pc. (42)	pc. (41)	pc. (42)	pc. (41)	pc. (42)	pc. (41)	pc. (42)
0	Min	D	7.5e7	3.6e6	7.5e7	3.3e6	3.1e6	1.6e4	1.1e7	3.9e4
	Max	D	3.9e13	1.2e20	1.1e13	5.5e19	1.7e10	2.6e20	4e10	1.2e18
	Min	S	5e7	9.8e7	5.3e7	8e7	3.7e6	4.2e5	7.1e6	1.4e6
	Max	S	2.5e13	2e21	6.7e12	3.3e21	7.4e9	6.6e20	2e10	3.2e19
0.005	Min	D	7.5e7	3.6e6	7.5e7	3.2e6	1.2e6	1.5e4	3.3e6	3.9e4
	Max	D	6.3e10	2.3e18	5.4e9	6.6e6	2.7e9	3.6e19	7.3e7	4.5e4
	Min	S	5e7	9.8e7	5.3e7	8e7	1.4e6	3.3e5	2.9e6	1.4e6
	Max	S	3.6e10	4.2e16	3.3e9	1.8e8	1.2e9	2.1e20	4.3e7	1.5e6
0.01	Min	D	7.5e7	3.6e6	7.5e7	3.2e6	1.2e6	1.5e4	3.3e6	3.9e4
	Max	D	2.2e10	3.8e15	2.3e9	6.6e6	7.1e8	2.9e17	2e7	4.5e4
	Min	S	5e7	9.8e7	5.3e7	8e7	1.4e6	3.3e5	3e6	1.4e6
	Max	S	1.3e10	4.1e16	1.5e9	1.8e8	3.3e8	3.8e17	1.4e7	1.5e6
0.05	Min	D	7.2e7	3.8e6	3e7	2.8e6	1.2e6	1.5e4	3.3e6	3.9e4
	Max	D	6.5e08	6.6e6	3.4e8	5.9e6	1.6e7	1.7e4	1.1e7	4.5e4
	Min	S	4.7e7	9e7	2.1e7	6.8e7	1.4e6	3.3e5	3e6	1.4e6
	Max	S	3.7e8	1.8e8	2.2e8	1.6e8	1.3e7	4.9e5	7.2e6	1.5e6
0.1	Min	D	5.2e7	3.5e6	3e7	2.8e6	1.2e6	1.5e4	3.3e6	3.9e4
	Max	D	2.3e8	5e6	1.5e8	4.6e6	5.1e6	1.7e4	1.1e7	4.5e4
	Min	S	3.7e7	8.8e7	2.1e7	6.7e7	1.4e6	3.3e5	3e6	1.4e6
	Max	S	1.4e8	1.5e8	9.4e7	1.3e8	6e6	4.9e5	7.2e6	1.5e6
0.2	Min	D	2.9e7	3.1e6	2.2e7	2.6e6	1.2e6	1.5e4	2.1e6	2.6e4
	Max	D	7.8e7	4.1e6	8.6e7	4.1e6	3.1e6	1.7e4	4.7e6	4.4e4
	Min	S	2e7	8.4e7	1.5e7	7e7	1.4e6	3.3e5	2.9e6	7.9e5
	Max	S	5.1e7	1.2e8	5.7e7	1.1e8	3.7e6	4.9e5	3.6e6	1.5e6

		cond.no.ofStokessystem :
This method :	$(k, k') = (2, 1)$	$2.79e5$
	$(k, k') = (3, 2)$	$5.83e5$
Hansbo *et al.* [14] :	$\text{P1} - \text{iso} - \text{P2}$	$6.74e5$

6. NUMERICAL RESULTS

For all two-phase problems that are studied later, the fluid interface $\mathfrak{I}$ is either circular or ellipsoidal. The level set function is given by

$$\varphi(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1, \tag{44}$$

which can be exactly represented in $\mathbb{P}_2(\mathfrak{K}_h)$. The cell agglomeration threshold for all examples in this section is set as $\alpha = 0.1$, and the block preconditioning matrix E is set as given by Equation (42). As already noted in the introduction, it is contradictory from a physical point-of-view to have a non-zero flow velocity and a stationary interface at the same time. These problems, however, are suitable mathematical model problems to study important numerical properties of the scheme, such as the convergence order.

A final comment is needed on the evaluation of the curvature: because we choose a level set function that can be represented exactly in the DG polynomial space and is in $\mathcal{C}^\infty(\Omega)$, the curvature can be exactly evaluated by Bonnet's formula. Therefore, the integral $\oint_{\mathfrak{I}} \kappa \sigma \vec{n}_{\mathfrak{I}} \text{dS}$ representing the force induced by surface tension can be evaluated with sufficient accuracy. For unsteady problems,

DOI: 10.1002/nme

Table II. For $\mu_{\mathfrak{A}} = \mu_{\mathfrak{B}} = 1$, the minimum and maximum condition number of the diffusion matrix $\mathcal{Q}_{\text{cd}}$ (table rows marked with D) and the full saddle point matrix $\mathcal{Q}$ (table rows marked with S) for all four different test configurations over various level set positions (Figure 4), for different agglomeration thresholds α and for two different block preconditioning options, as defined in Equations (41) and (42).

			Circle, shift test		Circle, rad. var. test		Parallel test		Diagonal test	
α			pc. (41)	pc. (42)	pc. (41)	pc. (42)	pc. (41)	pc. (42)	pc. (41)	pc. (42)
0	Min	D	2.1e5	1.1e5	2.1e5	1e5	9e4	1.6e4	3.7e5	3.4e4
	Max	D	6e10	2.2e19	4.6e10	5.3e18	5.5e8	2.1e17	3.6e9	4.5e15
	Min	S	5.4e9	8.3e4	5.4e9	8.3e4	3.3e6	1e4	1.7e7	1.9e4
	Max	S	7e16	1.4e19	9.6e14	6.4e18	2.3e10	1.9e18	3.5e11	5.8e16
0.005	Min	D	2.1e5	1.1e5	2.1e5	9.9e4	3.7e4	1e4	1.2e5	2.7e4
	Max	D	3.8e8	3e14	8e6	1.6e5	2.1e8	8.2e15	5.8e6	3.7e4
	Min	S	5.5e9	8.1e4	5.5e9	8.3e4	1.4e6	7.7e3	5.5e6	1.6e4
	Max	S	9.6e12	2.9e14	2e11	1.7e05	1.1e10	8.3e15	2.7e8	2.2e4
0.01	Min	D	2.1e5	1e5	1.6e5	9.9e4	3.7e4	1e4	1.2e5	2.7e4
	Max	D	8.1e7	3e14	8e6	1.6e5	5.4e7	2.8e14	1.6e6	3.6e4
	Min	S	5.4e9	8.2e4	4e9	8.3e4	1.3e6	7.4e3	5.5e6	1.6e4
	Max	S	2.1e12	2.9e14	2e11	1.7e5	2.4e9	2.8e14	7.2e7	2.2e4
0.05	Min	D	9.1e4	1e5	6.2e4	9.2e4	3.7e4	1e4	1.2e5	2.7e4
	Max	D	7.4e5	1.8e5	3.6e5	1.5e5	1.2e6	3.2e4	3.7e5	3.4e4
	Min	S	2.3e9	7.9e4	1.6e9	6.7e4	1.3e6	7.5e3	5.2e6	1.6e4
	Max	S	1.9e10	1.5e5	9.1e9	1.3e5	4.4e7	1.9e4	1.7e7	2e4
0.1	Min	D	9.1e4	1e5	5e4	9.1e4	3.7e4	1e4	1.2e5	2.7e4
	Max	D	2.6e5	1.4e5	1.6e5	1.2e5	3.7e5	2.2e4	3.7e5	3.4e4
	Min	S	2.3e9	6.5e4	1.3e9	6.3e4	1.3e6	7.4e3	5.5e6	1.6e4
	Max	S	6.6e9	1.2e5	4.1e9	9.8e4	1.4e7	1.3e4	1.7e7	2e4
0.2	Min	D	5.3e4	9.9e4	3e4	9.1e4	3.7e4	1e4	6.7e4	2.1e4
	Max	D	9.7e4	1.1e5	9.3e4	1.1e5	9e4	1.6e4	1.6e5	2.9e4
	Min	S	1.4e9	6.1e4	8e8	6.1e4	1.3e6	7.5e3	3e6	1.4e4
	Max	S	2.6e9	9.4e4	2.4e9	8.9e4	3.3e6	9e3	7e6	1.7e4

Table III. Maximum number of agglomerated cells for all three different test configurations (a circular level set, a level set parallel to the grid orientation, and a diagonal one) over various level set positions (Figure 6), for different agglomeration thresholds α and for two different block preconditioning options, as defined in Equations (41) and (42).

α	Circle, shift	Circle, rad. var.	Parallel	Diagonal
0	0	0	0	0
0.005	8	24	8	10
0.01	8	24	8	10
0.05	16	32	8	10
0.1	21	32	8	10
0.2	28	36	8	11

on the other hand, it cannot be assumed that $\varphi \in \mathcal{C}^\infty(\Omega)$; indeed, by using a DG-based algorithm for level set evolution, one cannot even assume $\varphi \in \mathcal{C}^0(\Omega)$. In such cases, a more sophisticated treatment of surface tension is required, to be investigated in future work.

In addition to the preprocessing presented in Section 5, Krylov methods would require additional preconditioning in order to solve the saddle point systems. A plain application of, for example, generalized minimal residual algorithm onto matrix $\mathcal{Q}$ (Equation (35)) does not converge without

DOI: 10.1002/nme

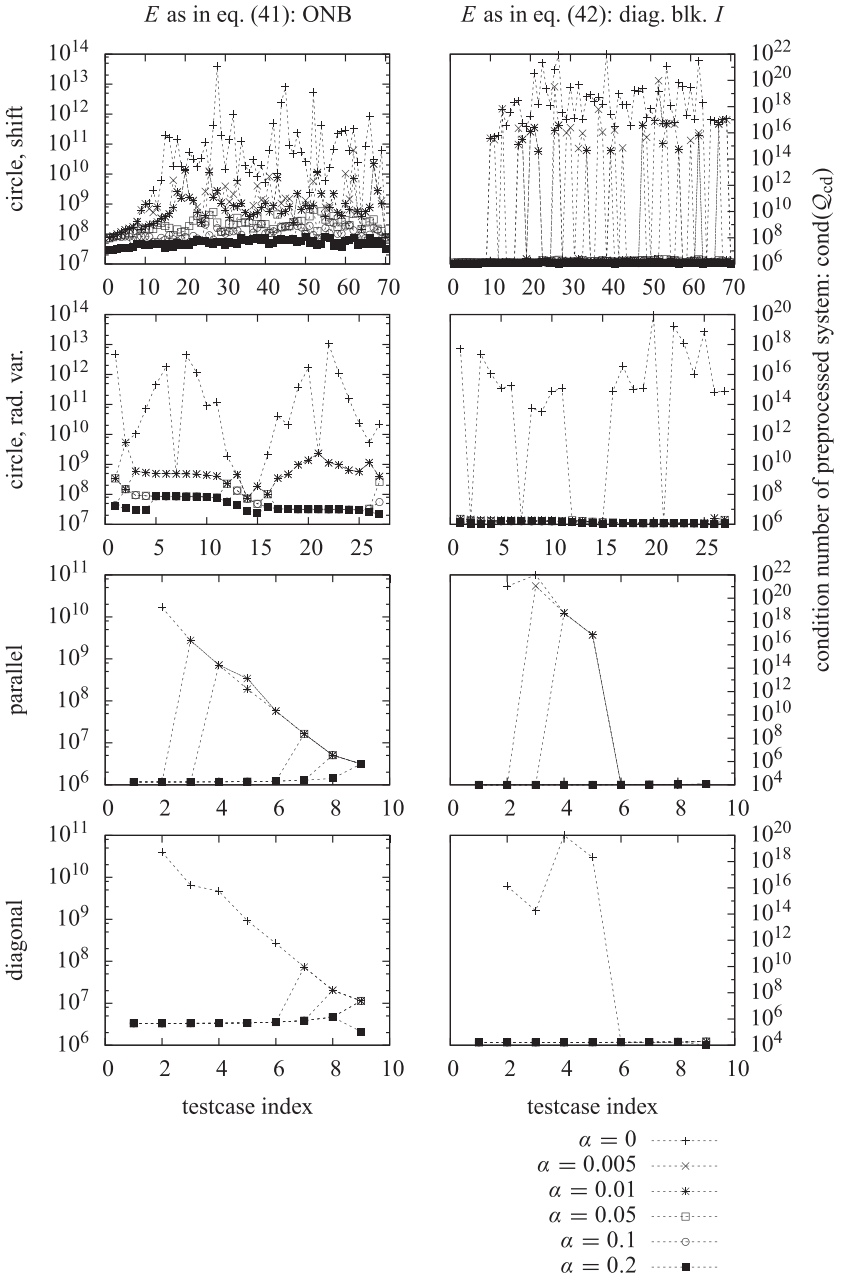


Figure 5. Condition number of the preprocessed diffusion system, that is, of the matrix $\mathcal{Q}_{\text{cd}}$ as defined in Equations (35) and (43), for various test cases and cell agglomeration thresholds α.

using a suitable preconditioner, for example, based on a Schur complement (e.g., [47, 48]). Therefore, in all tests given later, the direct solver PARDISO [44–46] is used to solve the discrete saddle point problem that arises from the XDG discretization of either the Stokes (Equation 18) or the Navier–Stokes (Equation 15) problem. For the rather small 2D problems presented herein, containing at most 2 331 084 DOFs, sparse direct solvers are usually faster than iterative methods. The use of a direct method is also convenient for the investigation of convergence, because any influence from the convergence criterion is eliminated.

Even if a direct solver is used, the method may still benefit from the preprocessing procedure. Especially, if low agglomeration thresholds α are used, the condition number of the saddle point matrix without preprocessing can be very high, far above 10^{12}. Under such circumstances,

DOI: 10.1002/nme

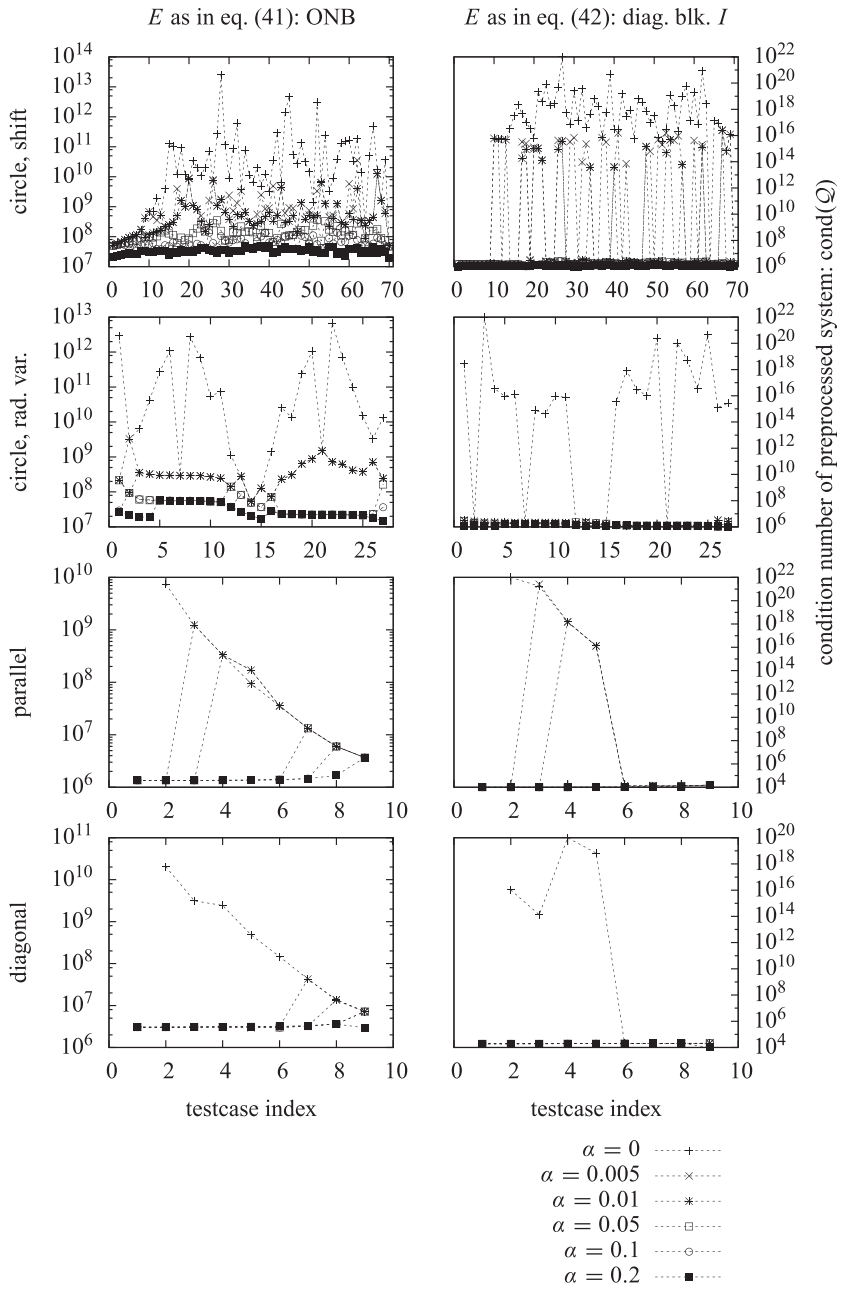


Figure 6. Condition number of the preprocessed saddle point system, that is, of the matrix $\mathcal{Q}$ as defined in Equation (35), for various test cases and cell agglomeration thresholds α.

we found PARDISO to be sometimes unreliable: If the same computation was performed several times, in approximately one out of five runs, PARDISO returned a wrong solution with a high residual. The reason may be that codes like PARDISO, which utilizes multithread-parallelization (OpenMP), are not completely deterministic because, for example, summation orders depend on the timing of threads. This is, however, somewhat speculative because the source code of PARDISO is not available. Here, the block preconditioning proves to be helpful in preventing the sparse solver from failing. This is surprising, because PARDISO should be able to perform the same equivalence transformation. The reason might be that PARDISO is not aware of the block structure induced by DG.

DOI: 10.1002/nme

6.1. A static droplet

First, a static droplet with a radius of 0.8 is investigated, that is, $a = b = 0.8$. The exact solution is

$$\vec{u}_{ex} = (0,0),\ p_{ex} = 0 \text{ in } \mathfrak{A} \text{ and } p_{ex} = \frac{-\sigma}{0.8} \text{ in } \mathfrak{B}.$$

This test case demonstrates the absence of 'spurious velocities', which have been reported to be an issue in other discretizations for two-phase flows and to illustrate the influence of the chosen HMF variant (Section 4). Here, the linearized Navier–Stokes problem is solved, that is, find $(\vec{u}, p) \in V_k^\alpha$ so that

$$Ns(\vec{u}_{ex}, (\vec{u}, p), (\vec{v}, \tau)) = rhsNs(\vec{v}, \tau) \quad \forall (\vec{v}, \tau) \in V_k^\alpha,$$

see also Equation (15). The material parameters for $\mathfrak{A}$ and $\mathfrak{B}$ are chosen to be those of water and air ($\sigma = 0.072$, $\rho_{\mathfrak{A}} = 10^3$, $\mu_{\mathfrak{A}} = 10^{-3}$ $\rho_{\mathfrak{B}} = 1.2$, $\mu_{\mathfrak{B}} = 17.1 \cdot 10^{-6}$). An 18×18 equidistant grid is chosen for the discretization of Ω, with an agglomeration factor $\alpha = 0.1$ and the DG polynomial degree $k = 3$. For the numerical solution $\vec{u}, p$ of the linearized Navier–Stokes problem, the following bounds can be obtained depending on the different HMF variants:

$\|\vec{u} - \vec{u}_{ex}\|_\infty$	$\|p - p_{ex}\|_\infty$	quadrature
$\leqslant 5.7 \cdot 10^{-6}$	$\leqslant 8.4 \cdot 10^{-5}$	$\mathrm{HMF_A}(8)$
$\leqslant 1.9 \cdot 10^{-9}$	$\leqslant 6.4 \cdot 10^{-7}$	$\mathrm{HMF_B}(8)$
$\leqslant 2.9 \cdot 10^{-10}$	$\leqslant 6.4 \cdot 10^{-9}$	$\mathrm{HMF_C}(8)$

It is remarkable that the result obtained by using $\mathrm{HMF_B}(8)$ is significantly more accurate than the one obtained by $\mathrm{HMF_A}(8)$, because the test carried out in Section 4.2 (see also Figure 3) indicated that the accuracy of $\mathrm{HMF_A}(-)$ and $\mathrm{HMF_B}(-)$ is almost the same. However, for the problem shown here, the difference in accuracy may be explainable by the fact that $\mathrm{HMF_B}(-)$ fulfills the Gauss theorem for the polynomial basis exactly up to machine accuracy on the discrete level, while $\mathrm{HMF_A}(-)$ does not.

We want to emphasize that even for lower approximation orders, that is, $k \leqslant 2$, it may be beneficial to use an accurate quadrature rule like HMF, depending on the required level of accuracy. For these approximation orders, it seems tempting to use, for example, a piecewise linear reconstruction of the interface $\mathfrak{I}$. This can be imitated by HMF by using a cell-wise linear, continuous approximation, that is, a continuous FEM, for φ instead of the quadratic space that was used to represent φ given by Equation (44) for all other examples in this work. For the radius of 0.8, the curvature obviously is equal to 1.25. However, the curvature field κ computed form φ is not polynomial, namely, $\kappa = \frac{1}{\sqrt{x^2+y^2}}$. Here, we use the exact curvature, and the only source of error is the cell-wise linear representation of $\mathfrak{I}$. In practice, one would also face the problem of how to recover the curvature form the cell-wise linear φ. Filter algorithms are discussed, for example, in [7]. Using an exact curvature on the wrong integration domain increases the error by several magnitudes:

k/k'	$\|\vec{u} - \vec{u}_{ex}\|_\infty$	$\|p - p_{ex}\|_\infty$	quadrature
1/0	$\leqslant 2.3 \cdot 10^{-10}$	$\leqslant 1.6 \cdot 10^{-12}$	$\mathrm{HMF_C}(8)$, quadratic φ
1/0	$\leqslant 6.7 \cdot 10^{-3}$	$\leqslant 6.6 \cdot 10^{-5}$	$\mathrm{HMF_C}(8)$, linear φ
2/1	$\leqslant 1.1 \cdot 10^{-10}$	$\leqslant 3.6 \cdot 10^{-10}$	$\mathrm{HMF_C}(8)$, quadratic φ
2/1	$\leqslant 2.3 \cdot 10^{-3}$	$\leqslant 2 \cdot 10^{-4}$	$\mathrm{HMF_C}(8)$, linear φ

6.2. Convergence study on the Stokes problem

We study the convergence order of the Stokes problem (Equation 18) on a sequence of meshes with

$$9 \times 9, 18 \times 18, 36 \times 36, \ldots, 576 \times 576$$

DOI: 10.1002/nme

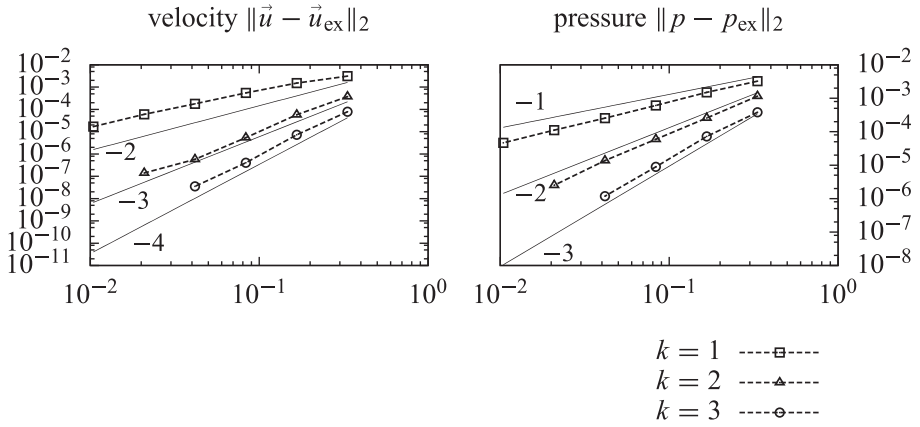


Figure 7. Convergence study for the ellipsoidal droplet in the L^2 norm; because no exact solution is known to this problem, the numerical solution on the finest grid was assumed for computing errors.

equidistant cells for the domain $\Omega = (-3/2, 3/2)$ with Dirichlet boundary conditions $\vec{u}_{\text{D}}$ on $\Gamma_{\text{D}} = \partial\Omega$. The parameters of the ellipse (44) and the physical parameters are set as

$$a = 0.816,\ b = 0.784,\ \mu_{\mathfrak{A}} = 0.5,\ \mu_{\mathfrak{B}} = 0.05,\ \sigma = 0.1.$$

Because we are not aware of an exact solution to this problem, we assume $(\vec{u}_{ex}, p_{ex})$ to be the solution observed on the finest grid. Solutions on grids with coarser resolutions are injected onto the finest grid in order to compute L^2 errors. As an average experimental order of convergence (EOC) we observe

k/k'	EOC velocity	EOC pressure	grids	'ex. sol.' grid
1/0	1.52	1.23	$9 \times 9 - 288 \times 288$	576×576
2/1	2.91	2.19	$9 \times 9 - 144 \times 144$	288×288
3/2	3.74	2.90	$9 \times 9 - 72 \times 72$	144×144

The trend can also be seen in Figure 7, which approximately matches the expected behavior – convergence orders of $k + 1$ for velocity and k for pressure – with exception of the $k/k' = 1/0$ - case. For an exemplary plot of the solution behavior with viscosities of air and water, see Figure 8.

6.3. A steady Navier–Stokes solution: two-phase Taylor–Couette flow

Finally, we demonstrate the solution of a steady Navier–Stokes problem. A solution $(\vec{u}, p)$ to the nonlinear problem (15) is obtained as the limit of an iterative sequence

$$(\vec{u}_0, p_0) = 0, (\vec{u}_1, p_1), (\vec{u}_2, p_2), \ldots \to (\vec{u}, p). \tag{45}$$

For $\vartheta = 1, 2, \ldots$, the intermediate solution $(\vec{u}_{\vartheta+1}, p_{\vartheta+1})$ is found by a fix point iteration:

$$\text{Find } (\vec{u}_{\vartheta+1}, p_{\vartheta+1}) \in V_p^\alpha \text{ s. t.} \quad Ns(\vec{u}_\vartheta, (\vec{u}_{\vartheta+1}, p_{\vartheta+1}), (\vec{v}, \tau)) = rhsNs((\vec{v}, \tau)) \quad \forall (\vec{v}, \tau) \in V_k^\alpha. \tag{46}$$

The Taylor–Couette flow is the flow between two rotating cylinders in 3D with outer radius r_a and inner radius r_i. In the two-phase case, the fluid interface is a circle of radius r_m, so $r_i < r_m < r_a$, and for the level set function (44), the factors a and b are set $a = b = r_m$. So, in 2D, the domain is $\Omega = \{\vec{x} \in \mathbb{R}^2;\ r_i < |\vec{x}|_2 < r_a\}$ with boundary conditions

$$\vec{u}_{\text{D}} = \frac{(-y, x)^T}{\sqrt{x^2 + y^2}} \cdot \begin{cases} u_i & \text{for } x^2 + y^2 = r_i^2 \\ u_a & \text{for } x^2 + y^2 = r_a^2 \end{cases}. \tag{47}$$

DOI: 10.1002/nme

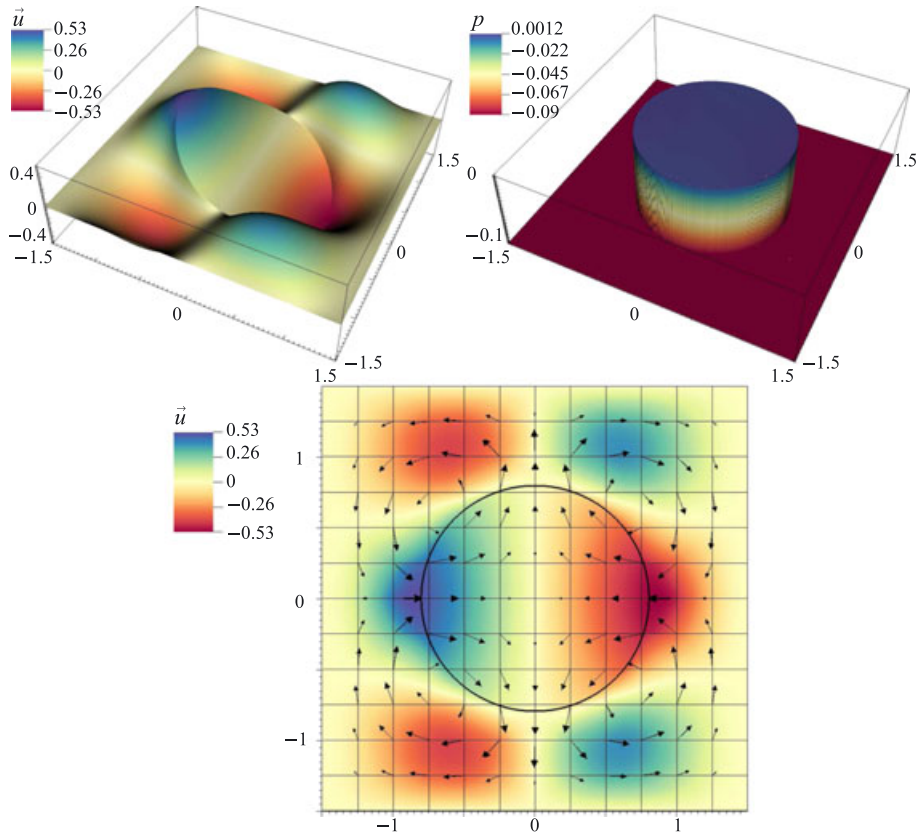


Figure 8. Velocity in x-direction (left side), pressure (right side), and velocity vectors/interface position/grid (bottom) for the steady ellipsoidal droplet. It can be seen that kinks in the velocity field and jumps in the pressure field can be represented nicely, even on a rather coarse grid of 12×12 cells for the domain $\Omega = (-3/2, 3/2)^2$. The material parameters are chosen to be those of air and water, that is, $\sigma = 0.072$, $\mu_{\mathfrak{A}} = 10^{-3}$, $\mu_{\mathfrak{B}} = 17.1 \cdot 10^{-6}$, and the parameter for the ellipse are $a = 0.8\sqrt{1.01} \approx 0.80399$ and $b = 0.8\sqrt{0.99} \approx 0.79599$ (Equation (44)).

For parameters

$$r_i = \sqrt{2}/2, r_a = 2, r_m := (r_i + r_a)/2, u_i = 2, u_a = 1,$$
$$\rho_{\mathfrak{A}} = 0.1, \rho_{\mathfrak{B}} = 1.3, \mu_{\mathfrak{A}} = 0.01, \mu_{\mathfrak{B}} = 0.2, \sigma = 0.9$$

one obtains the exact solution

$$\vec{u}_{ex} = \frac{(-y, x)^T}{\sqrt{x^2 + y^2}} u^r_{ex}\left(\sqrt{x^2 + y^2}\right) \text{ and } p_{ex} = \frac{1}{\sqrt{x^2 + y^2}} p^r_{ex}\left(\sqrt{x^2 + y^2}\right) \tag{48}$$

with

$$\begin{aligned}
u^r_{ex}(r) &= 1.585073289/r - 0.3417194536r && \text{for } r < r_m,\\
u^r_{ex}(r) &= 0.07925366446/r + 0.480186584r && \text{for } r > r_m,\\
p^r_{ex}(r) &= 0.005838609245r^2 - 0.1083300757 \ln r\\
&\quad - 0.1256228666/r^2 + 1.057897889 && \text{for } r < r_m,\\
p^r_{ex}(r) &= 0.1498764510r^2 + 0.09894702069 \ln r\\
&\quad - 0.004082743166/r^2 && \text{for } r > r_m.
\end{aligned}$$

DOI: 10.1002/nme

This solution is plotted in Figure 9. A DG discretization of the ring-typed domain Ω described previously would require the use of higher-order curved elements in order to obtain the desired convergence rates for $k > 0$. Unfortunately, curved elements are not available in our present XDG implementation. However, because the solution (48,6.3) is continuable outside of the singular point $(0, 0)$, we choose the domain

$$\Omega = (-2, 2)^2 \setminus (-1/2, 1/2)^2,$$

which is discretized by equidistant Cartesian grids with

$$8 \times 8,\ 16 \times 16,\ 32 \times 32, \ldots, 512 \times 512$$

cells. The exact solution (48,6.3) is used as a Dirichlet boundary condition, where $\Gamma_D = \partial\Omega$. We observe the following average EOC in the L^2 norm:

k/k'	EOC velocity	EOC pressure	grids
1/0	1.71	1.54	$8 \times 8 - 512 \times 512$
2/1	3.28	2.17	$8 \times 8 - 256 \times 256$
3/2	4.04	2.81	$8 \times 8 - 128 \times 128$

The trend can also be seen in Figure 10, which approximately matches the expected behavior with exception of the $k/k' = 1/0$ case. For all calculations, the iterative sequence ((45) and (46)) was executed for about 20 iterations; already after about 10 iterations, no further reduction of the residual of problem (46) could be observed. An exemplary plot of the residual convergence is shown in Figure 11.

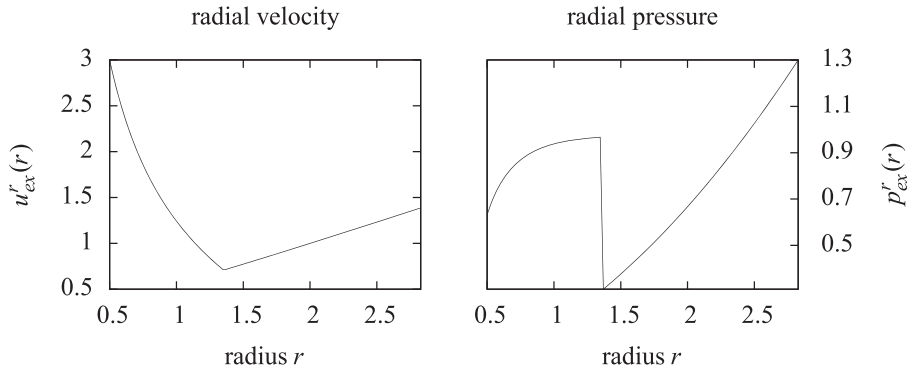


Figure 9. Exact solution for the two-phase Taylor–Couette flow, in dependence of the radial coordinate $r = \sqrt{x^2 + y^2}$.

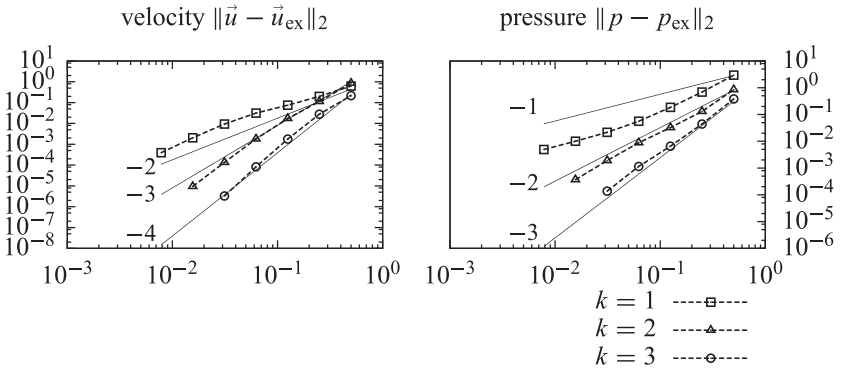


Figure 10. Convergence study for the two-phase Taylor–Couette flow in the L^2 norm.

DOI: 10.1002/nme

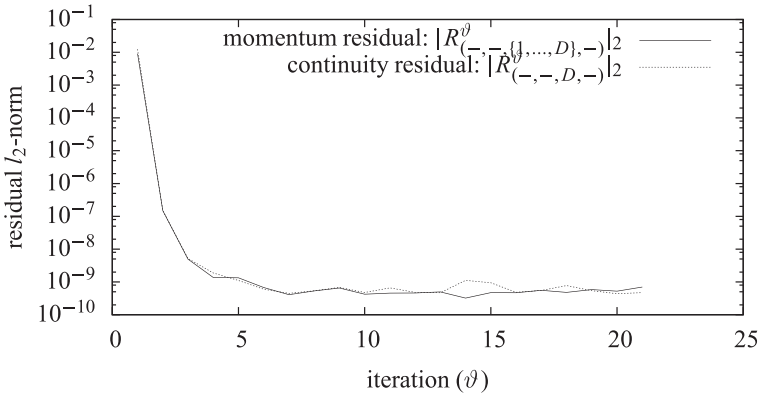


Figure 11. Convergence history of the velocity and pressure residual over the first 20 iterations of the fix-point iteration (45) and (46) on the 128×128 grid, for $k/k' = 3/2$. The residual vector, R^{ϑ}, in the ϑ-th iteration is defined as $R^{\vartheta}_i := rhsNs(\theta_i) - Ns(\vec{u}^{\vartheta}, (\vec{u}^{\vartheta}, p^{\vartheta}), \theta_i)$, for a basis $\underline{\theta} = (\theta_1, \dots \theta_I)$ of the agglomerated space V^{α}_k. For details on the multi-index notation on R^{ϑ}, by which momentum and continuity components are selected and the basis $\underline{\theta} = \underline{\phi}BCE$, see Section 5, especially Definition 6 and Equation (35). The block preconditioning matrix E is chosen as defined by Equation (42).

7. SUMMARY AND OUTLOOK

Within this work, the spatial discretization of a two-phase flow problem by means of a level set/extended DG discretization has been demonstrated. The numerical results show a promising level of accuracy, especially on low-resolution grids, and indicate that the convergence properties of the DG method can be recovered in a two-phase setting by the XDG extension. Furthermore, cell agglomeration seems to be an appropriate tool to handle arbitrary cut situations.

Several open topics remain to be resolved: Of primary importance will be the investigation of dynamic problems in two and three dimensions. Although not discussed here, the spatial discretization shows a high sensitivity to errors on the curvature. This issue has already been addressed by [7]. Preliminary results indicate that a level set evolution algorithm, where errors in the curvature do not accumulate in time, is required.

Furthermore, the use of a direct sparse solver is not feasible for computations far above 10^6 DOF. The largest calculation presented herein contains 2 331 084 DOF (at $k = 1$, on the 576×576 grid, see Section 6.2), which is about the practical limit of the PARDISO solver on a conventional desktop computer. Therefore, future work will also aim at efficient preconditioning and iterative solvers for those systems.

ACKNOWLEDGEMENTS

This project is funded by the German Research Association (Deutsche Forschungsgemeinschaft, DFG) under fund KU-2719/1-1.
I would like to acknowledge the support of Prof. Parviz Moin and the CTR department at Stanford University in hosting this research project. I also thank Prof. Tim Warburton for fruitful and challenging discussions, as well as for his support for my visit at the CAAM department at Rice University, Houston, TX, USA.

REFERENCES

1. Olsson E, Kreiss G. A conservative level set method for two phase flow. *Journal of Computational Physics* 2005; **210**(1):225–246.
2. Babuška I. The finite element method for elliptic equations with discontinuous coefficients. *Computing* 1970; **5**(3):207–213.
3. Barrett JW, Elliott CM. Fitted and unfitted finite-element methods for elliptic equations with Smooth interfaces. *IMA Journal of Numerical Analysis* 1987; **7**(3):283–300.

DOI: 10.1002/nme

4. Moës N, Dolbow J, Belytschko T. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering* 1999; **46**:131–150.
5. Müller B, Kummer F, Oberlack M. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *International Journal for Numerical Methods in Engineering* 2013; **96**(8):512–528.
6. Bastian P, Engwer C. An unfitted finite element method using discontinuous Galerkin. *International Journal for Numerical Methods in Engineering* 2009; **79**(12):1557–1576.
7. Kummer F, Warburton T. Patch-recovery filters for curvature in discontinuous Galerkin-based level-set methods. *Communications in Computational Physics.* 2016; **19**(02):329–353.
8. Chessa J, Belytschko T. Arbitrary discontinuities in space–time finite elements by level sets and X-FEM. *International Journal for Numerical Methods in Engineering* 2004; **61**(15):2595–2614.
9. Gross S, Reusken A. An extended pressure finite element space for two-phase incompressible flows with surface tension. *Journal of Computational Physics* 2007; **224**(1):40–58.
10. Gross S, Reusken A. Finite element discretization error analysis of a surface tension force in two-phase incompressible flows. *SIAM Journal on Numerical Analysis* 2007; **45**(4):1679–1700.
11. Fries TP. The intrinsic XFEM for two-fluid flows. *International Journal for Numerical Methods in Fluids* 2009; **60**(4):437–471.
12. Cheng K-W, Fries T-P. XFEM with hanging nodes for two-phase incompressible flow. *Computer Methods in Applied Mechanics and Engineering* 2012; **245-246**:290–312. DOI: 10.1016/j.cma.2012.07.011.
13. Sauerland H, Fries T-P. The stable XFEM for two-phase flows. *Computers & Fluids* 2013; **87**:41–49. USNCCM Moving Boundaries.
14. Hansbo P, Larson MG, Zahedi S. A cut finite element method for a Stokes interface problem. *Applied Numerical Mathematics* 2014; **85**:90–114. DOI: 10.1016/j.apnum.2014.06.009.
15. Burman E. Ghost penalty. *Comptes Rendus Mathematique* 2010; **348**(21-22):1217–1220.
16. Reed WH, Hill TR. Triangular mesh methods for the neutron transport equation. *National Topical Meeting on Mathematical Models and Computational Techniques for Analysis of Nuclear Systems*, CONF-730414–2; LA-UR–73-479, Los Alamos Scientific Lab., N.Mex. (USA), 1973; 23.
17. Di Pietro DA, Ern A. *Mathematical Aspects of Discontinuous Galerkin Methods*, No. 69 in Mathématiques et Applications. Springer: Heidelberg, 2011.
18. Karniadakis G, Sherwin SJ. *Spectral/hp Element Methods for Computational Fluid Dynamics* (2nd ed), Numerical mathematics and scientific computation. Oxford University Press: New York, 2005.
19. Cockburn B, Dong B, Guzmán J. A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems. *Mathematics of Computation* 2008; **77**(264):1887–1916.
20. Kirby RM, Sherwin SJ, Cockburn B. To CG or to HDG: a comparative study. *Journal of Scientific Computing* 2012; **51**(1):183–212.
21. Huerta A, Roca X, Aleksandar A, Peraire J. Are high-order and hybridizable discontinuous Galerkin methods competitive? *Oberwolfach Reports* 2012; **9**(1):485–487.
22. Huerta A, Angeloski A, Roca X, Peraire J. Efficiency of high-order elements for continuous and discontinuous Galerkin methods. *International Journal for Numerical Methods in Engineering* 2013; **96**(9):529–560.
23. Heimann F, Engwer C, Ippisch O, Bastian P. An unfitted interior penalty discontinuous Galerkin method for incompressible Navier–Stokes two-phase flow. *International Journal for Numerical Methods in Fluids* 2013; **71**(3): 269–293.
24. Massjung R. An unfitted discontinuous Galerkin method applied to elliptic interface problems. *SIAM Journal on Numerical Analysis* 2012; **50**(6):3134–3162.
25. Massing A, Larson MG, Logg A, Rognes ME. A stabilized Nitsche overlapping mesh method for the Stokes problem. *Numer. Math.* 2014; **128**(1):73–101.
26. Annavarapu C, Hautefeuille M, Dolbow JE. A robust Nitsche's formulation for interface problems. *Computer Methods in Applied Mechanics and Engineering* 2012; **225-228**:44–54. DOI: 10.1016/j.cma.2012.03.008.
27. Saye RI. High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles. *SIAM Journal on Scientific Computing* 2015; **37**(2):A993–A1019.
28. Hesthaven JS, Warburton T. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, No. 54 in Texts in Applied Mathematics. Springer-Verlag: Heidelberg, 2008.
29. Sethian JA. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press: Cambridge, 1996.
30. Osher SJ, Fedkiw RP. *Level Set Methods and Dynamic Implicit Surfaces*. Springer: New York, 2002.
31. Babuška I. The finite element method with Lagrangian multipliers. *Numerische Mathematik* 1973; **20**(3):179–192.
32. Brezzi F. On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique* 1974; **8**(R2):129–151.
33. Girault V, Rivière B, Wheeler MF. A discontinuous Galerkin method with nonoverlapping domain decomposition for the Stokes and Navier–Stokes problems. *Mathematics of Computation* 2005; **249**:53–84.
34. Klein B, Kummer F, Oberlack M. A SIMPLE based discontinuous Galerkin solver for steady incompressible flows. *Journal of Computational Physics* 2013; **237**:235–250.
35. Shahbazi K, Fischer P, Ethier RC. A high-order discontinuous Galerkin method for the unsteady incompressible Navier–Stokes equations. *Journal of Computational Physics* 2007; **222**(1):391–407.

36. Arnold DN. An interior penalty finite element method with discontinuous elements. *SIAM Journal on Numerical Analysis* 1982; **19**(4):742–760.
37. Arnold DN, Brezzi F, Cockburn B, Marini LD. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis* 2002; **39**(5):1749–1779.
38. Shahbazi K. An explicit expression for the penalty parameter of the interior penalty method. *Journal of Computational Physics* 2005; **205**:401–407.
39. Warburton T, Hesthaven JS. On the constants in hp-finite element trace inverse inequalities. *Computer Methods in Applied Mechanics and Engineering* 2003; **192**(25):2765–2773.
40. Burman E, Ern A. Continuous interior penalty hp-finite element methods for advection and advection–diffusion equations. *Mathematics of Computation* 2007; **76**(259):1119–1140.
41. Hillewaert K. Development of the discontinuous Galerkin method for high-resolution, large scale CFD and acoustics in industrial geometries. *PhD thesis*, Université catholique de Louvain, Louvain-la-Neuve, Belgium.
42. Antonietti PF, Giani S, Houston P. hp-version composite discontinuous Galerkin methods for elliptic problems on complicated domains. *SIAM Journal on Scientific Computing* 2013; **35**(3):A1417–A1439.
43. Cangiani A, Georgoulis EH, Houston P. hp-version discontinuous Galerkin methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences* 2014; **24**(10):2009–2041.
44. Schenk Olaf. Two-level dynamic scheduling in PARDISO: improved scalability on shared memory multiprocessing systems. *Parallel Computing* 2002; **28**(2):187–197.
45. Schenk O. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems* 2004; **20**(3):475–487.
46. Schenk O, Gärtner K. On fast factorization pivoting methods for sparse symmetric indefinite systems. *Electronic Transactions on Numerical Analysis* 2006; **23**:158–179.
47. Elman HC, Golub GH. Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM Journal on Numerical Analysis* 1994; **31**(6):1645–1661.
48. Benzi M, Golub GH, Liesen J. Numerical solution of saddle point problems. *Acta Numerica* 2005; **14**:1–137. DOI: 10.1017/S0962492904000212.