INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2017; **110**:3–30
Published online 23 November 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.5343

A high-order discontinuous Galerkin method for compressible flows with immersed boundaries

B. Müller[1,2,*,†], S. Krämer-Eis[1,2], F. Kummer[1,2] and M. Oberlack[1,2,3]

[1] *Chair of Fluid Dynamics, Technische Universität Darmstadt, 64287 Darmstadt, Germany*
[2] *Graduate School of Computational Engineering, Technische Universität Darmstadt, 64293 Darmstadt, Germany*
[3] *Center of Smart Interfaces, Technische Universität Darmstadt, 64287 Darmstadt, Germany*

SUMMARY

We present a higher order discretization scheme for the compressible Euler and Navier–Stokes equations with immersed boundaries. Our approach makes use of a discontinuous Galerkin discretization in a domain that is implicitly defined by means of a level set function. The zero iso-contour of this level set function is considered as an additional domain boundary where we weakly enforce boundary conditions in the same manner as in boundary-fitted cells. In order to retain the full order of convergence of the scheme, it is crucial to perform volume and surface integrals in boundary cells with high accuracy. This is achieved using a linear moment-fitting strategy. Moreover, we apply a non-intrusive cell-agglomeration technique that averts problems with very small and ill-shaped cuts. The robustness, accuracy, and convergence properties of the scheme are assessed in several two-dimensional test cases for the steady compressible Euler and Navier–Stokes equations. Approximation orders range from 0 to 4, even though the approach directly generalizes to even higher orders. In all test cases with a sufficiently smooth solution, the experimental order of convergence matches the expected rate for discontinuous Galerkin schemes. Copyright © 2016 John Wiley & Sons, Ltd.

Received 6 April 2016; Revised 30 June 2016; Accepted 29 July 2016

KEY WORDS: discontinuous Galerkin method; immersed boundary; level set; quadrature; compressible flow; cell-agglomeration

1. INTRODUCTION

The term immersed boundary method (IBM) has been coined by Peskin [1] for the simulation of blood flow on a Cartesian grid whose boundary is not aligned with the considered geometry. Numerous improvements and formulations for various discretization strategies have been proposed and applied successfully [2], especially in configurations with complicated or moving geometries. However, almost all of these variants share the disadvantage that they are limited to second-order accuracy in the best case.

One of the few exceptions is the recent work by Qin and Krivodonova [3] who present numerical results for a time-explicit discontinuous Galerkin (DG) discretization of the Euler equations with up to third-order convergence rates. In their approach, the immersed boundary is given by the zero iso-contour of a level set function, and all intersected cells are split into piecewise planar quadrature sub-cells. The authors then merge small or ill-shaped cut cells and apply a modified boundary condition for adiabatic slip walls [4] on each of these planar sections in order to retain the full convergence rate of the DG scheme.

Within this work, we present a related approach that generalizes the work of Qin and Krivodonova [3] in three important aspects. First, we use a different strategy for the numerical integration in cells

*Correspondence to: B. Müller, Chair of Fluid Dynamics, Technische Universität Darmstadt, 64287 Darmstadt, Germany.
†E-mail: mueller@fdy.tu-darmstadt.de

intersected by the immersed boundary, which allows us to avoid integration sub-cells and, as a result, to extend the scheme to higher approximation orders. Second, we do not require any reformulation of the boundary conditions, which allows us to evade the limitation to adiabatic slip walls. Finally, we have extended the approach to the compressible Navier–Stokes equations.

1.1. State of the art

Methods in the spirit of the original IBM by Peskin [1] have drawn much interest ever since their introduction in 1972 [2]. Within the present work, the term IBM is used in the general sense of methods making use of computational domains that do not conform with the problem domain, hence separating the aspects of discretization and geometry representation to a large extent. This idea is the core of many modern developments in the context of the finite element method such as the extended finite element method (XFEM) [5], the finite cell method [6], Nitsche-type methods [7], and even the finite difference method [8]. In the following, we will focus on numerical methods of the finite element type that are able to deliver higher order convergence rates in the presence of curved immersed problem geometries.

First efforts in this direction using XFEM/Nitsche-type approaches have been presented in [9, 10] in the context of linear elasticity and in [11] in the context of Stokes flow. Both approaches rely on planar surface triangulations for the numerical integration of the weak forms, which renders the introduction of quadrature sub-cells inevitable if higher order convergence of the scheme is desired. An interesting alternative has been proposed in [12] where special enrichment functions for common features such as circular sections or corners are introduced. Unfortunately, the presented scheme appears to be limited to second-order accuracy.

The development of a DG IBM has first been reported by Fidkowski and Darmofal [13] where the authors study the implicit solution of steady compressible flows over two-dimensional geometries based on an h-adaptive cut cell strategy. Regarding numerical integration in cut cells, the paper contains first efforts to apply moment-fitting-type numerical integration for complicated cut geometries (cf. Section 4), even though the authors rely on a spline-based reconstruction of the embedded geometry and an exceedingly large number of quadrature nodes. The latter deficiency has been relieved in follow-up works [14, 15] that also discuss a cell-agglomeration strategy to improve conditioning of the system matrix. Some of the ideas presented in these works have been reused in the aforementioned work by Qin and Krivodonova [3] who use an explicit time-integrator to drive the Euler equations into a steady-state.

A larger number of publications is concerned with higher order solutions of Poisson-type problems. To the best of our knowledge, the first notable contribution has been presented by Bastian and Engwer [16] who study solutions in porous media using an unfitted DG scheme. Later, this work has been extended to incompressible two-phase flows [17]. Other authors have used the aforementioned Nitsche-type methods [18] inspired by the ideas presented in [19]. All these methods share the common idea of using local, piecewise planar sub-triangulations of boundary cells for the numerical integration of the weak forms.

Recently, related methods with alternative approaches to numerical integration have been proposed in [20] and [21] for the Poisson equation and by [22] for steady incompressible two-phase flows. We will discuss these works separately in Section 4 with a focus on quadrature in cut cells.

1.2. Outline

This work is structured as follows: In Section 2, we recall the compressible Euler and Navier–Stokes equations in a dimensionless, conservative form, before introducing the baseline DG discretization with immersed boundaries in Section 3. Two major ingredients of the scheme, namely, the numerical integration in cut cells and the cell-agglomeration strategy, are discussed in detail in Sections 4 and 5, respectively. Finally, we assess the robustness, accuracy, and convergence behavior of the discretization in Section 6 using several two-dimensional test cases for the steady compressible Euler and Navier–Stokes equations.

DOI: 10.1002/nme

2. GOVERNING EQUATIONS

We consider the two-dimensional Navier–Stokes equations in a dimensionless, conservative form:

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}_1^c(\vec{U})}{\partial x} + \frac{\partial \vec{F}_2^c(\vec{U})}{\partial y} - \frac{\partial \vec{F}_1^v(\vec{U}, \nabla \vec{U})}{\partial x} - \frac{\partial \vec{F}_2^v(\vec{U}, \nabla \vec{U})}{\partial y} = 0, \tag{1}$$

supplemented by suitable initial and boundary conditions. For $d = 1, 2$, the vectors of conservative variables $\vec{U}$, convective fluxes $\vec{F}_d^c$, and viscous fluxes $\vec{F}_d^v$ are given

$$\vec{U} = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{pmatrix}, \tag{2}$$

$$\vec{F}_d^c = \begin{pmatrix} \rho u_d \\ \rho u_d u_1 + p\delta_{d,1} \\ \rho u_d u_2 + p\delta_{d,2} \\ u_d (\rho E + p) \end{pmatrix}, \tag{3}$$

and

$$\vec{F}_d^v = \frac{1}{\mathrm{Re}} \begin{pmatrix} 0 \\ \tau_{d,1} \\ \tau_{d,2} \\ \tau_{d,1} u_1 + \tau_{d,2} u_2 + \frac{\gamma}{\mathrm{Pr}(\gamma-1)} q_d \end{pmatrix}, \tag{4}$$

respectively, where ρ is the density, u_d is the d-th component of the velocity vector $\vec{u}$, ρE is the total energy per unit mass, p is the pressure, τ is the stress tensor, q is the heat flux, and δ is the Kronecker delta. The dimensionless parameters are given by the heat capacity ratio γ, the Reynolds number Re, and the Prandtl number Pr. Throughout this work, we set $\gamma = 1.4$ to model standard air.

The system is closed using the ideal gas law

$$p = (\gamma - 1)\rho e, \tag{5}$$

where e is the specific inner energy. Moreover, we assume a Newtonian fluid with viscous stresses defined by

$$\tau = \mu \left[\nabla \vec{u} + \nabla \vec{u}^T - \frac{2}{3} (\nabla \cdot \vec{u}) \mathbf{I} \right], \tag{6}$$

where the dynamic viscosity μ is taken as constant and $\mathbf{I}$ is the identity matrix.

The heat fluxes are modeled using Fourier's law,

$$q_d = k \frac{\partial T}{\partial x_d}, \tag{7}$$

where T and k denote the temperature and thermal conductivity, respectively. Assuming a constant Prandtl number, the thermal conductivity is given by

$$k = \mu. \tag{8}$$

DOI: 10.1002/nme

3. DISCRETIZATION

In the following, we briefly outline the standard DG discretization of a generic conservation law, before discussing the differences to our implementation of a DG IBM scheme in Section 3.2. The most important building blocks for this scheme are the numerical integration in cut cells and a cell-agglomeration strategy, which will be discussed in Sections 4 and 5, respectively.

3.1. Standard discontinuous Galerkin discretization

For the sake of simplicity, we will introduce the basic form of the DG method using the example of the D-dimensional scalar conservation law

$$\frac{\partial c}{\partial t} + \nabla \cdot \vec{f}(c) = 0 \tag{9}$$

for some concentration $c = c(\vec{x}, t)$ with $\vec{x} \in \Omega \subset \mathbb{R}^D$, $t \in \mathbb{R}_0^+$ and a smooth function $\vec{f} : \mathbb{R} \to \mathbb{R}^D$, supplemented with suitable initial and boundary conditions. The extension to the Euler equations is straightforward, while viscous terms in the Navier–Stokes equations need special attention. Here, we directly follow the approach presented in [23, Section 3], which is why we omit the details at this point and defer the discussion of the penalty parameter to Section 6.

Let Ω_h be a discretization of Ω with a characteristic mesh parameter h that represents a measure for the size of the affine linear cells $\{\mathcal{K}_i\}_{i=1,\dots,N}$ forming a tessellation of Ω_h. Within this setting, consider the set of polynomial trial and test functions $\{\Phi_{i,j}\}_{j=1,\dots,M}$ of maximum degree P, where

$$\operatorname{supp}\left(\Phi_{i,j}\right) = \mathcal{K}_i. \tag{10}$$

For each cell $\mathcal{K}_i$ with outward unit normal vector $\vec{n}_i$, the cell-local DG basis is then given by the row vector $\vec{\Phi}_i = (\Phi_{i,1}, \dots, \Phi_{i,M})$.

We multiply Equation (9) by $\Phi_{i,j}$, integrate over the cell $\mathcal{K}_i$, and perform an integration by parts in order to obtain

$$\int_{\mathcal{K}_i} \frac{\partial c}{\partial t} \Phi_{i,j}\, dV + \int_{\partial\mathcal{K}_i} \left(\vec{f}(c) \cdot \vec{n}_i\right) \Phi_{i,j}\, dA - \int_{\mathcal{K}_i} \vec{f}(c) \cdot \nabla \Phi_{i,j}\, dV = 0. \tag{11}$$

We choose to discretize c by means of the modal approximation

$$c(\vec{x}, t)\big|_{\mathcal{K}_i} \approx \tilde{c}(\vec{x}, t)\big|_{\mathcal{K}_i} = \tilde{c}_i(\vec{x}, t) = \sum_{k=1}^{M} \Phi_{i,k}(\vec{x})\, \tilde{c}_{i,k}(t) \tag{12}$$

with the column vectors of coefficients $\vec{\tilde{c}}_i = \vec{\tilde{c}}_i(t) = (\tilde{c}_{i,1}(t), \dots, \tilde{c}_{i,M}(t))^{\mathrm{T}}$, viz.

$$\tilde{c}_i(\vec{x}, t) = \vec{\Phi}_i \vec{\tilde{c}}_i. \tag{13}$$

Let the edges of $\mathcal{K}_i$ be denoted by $\varepsilon_{i,1}, \dots, \varepsilon_{i,E}$. Inserting (12) into (11) and replacing $\vec{f}(c) \cdot \vec{n}_i$ by a consistent numerical flux $g(\tilde{c}_i^-, \tilde{c}_i^+, \vec{n}_i)$ in the surface integral in (11) lead to the local form

$$\int_{\mathcal{K}_i} \frac{\partial \tilde{c}_i}{\partial t} \Phi_{i,j}\, dV + (\vec{g}_i)_j = 0 \tag{14}$$

of a semi-discrete set of equations with

$$(\vec{g}_i)_j := \sum_{e=1}^{E} \int_{\varepsilon_{i,e}} g\left(\tilde{c}_i^-, \tilde{c}_i^+, \vec{n}_i\right) \Phi_{i,j}\, dA - \int_{\mathcal{K}_i} \vec{f}(\tilde{c}_i) \cdot \nabla \Phi_{i,j}\, dV. \tag{15}$$

Here,

$$\tilde{c}_i^- = \tilde{c}_i^-(\vec{x}, t) := \lim_{\epsilon \to 0^+} \tilde{c}(\vec{x} - \epsilon \vec{n}_i, t) \tag{16}$$

DOI: 10.1002/nme

and

$$\tilde{c}_i^+ = \tilde{c}_i^+(\vec{x},t) := \begin{cases} \tilde{c}^{\mathrm{bc}} & \varepsilon_{i,e} \subset \partial\Omega_h \\ \lim\limits_{\epsilon\to 0^+} \tilde{c}(\vec{x}+\epsilon\vec{n}_i,t) & \text{otherwise} \end{cases} \tag{17}$$

are the one-sided limits of the approximate solution on $\varepsilon_{i,e}$, where $\tilde{c}_i^+$ depends on the boundary value $\tilde{c}^{\mathrm{bc}} = \tilde{c}^{\mathrm{bc}}(\vec{x},t,\tilde{c})$ if $\varepsilon_{i,e}$ is a boundary edge.

The temporal term in Equation (14) can be simplified to

$$\int_{\mathcal{K}_i} \frac{\partial \tilde{c}_i}{\partial t}\Phi_{i,j}\,dV = \sum_{k=1}^{M} \frac{\partial \tilde{c}_{i,k}}{\partial t} \underbrace{\int_{\mathcal{K}_i} \Phi_{i,k}\Phi_{i,j}\,dV}_{=:(\mathbf{M}_i)_{k,j}}, \tag{18}$$

where $\mathbf{M}_i \in \mathbb{R}^{M,M}$ denotes the cell-local, symmetric mass matrix of $\mathcal{K}_i$. It directly follows that the semi-discrete system can be summarized as follows:

$$\mathbf{M}_i \frac{\partial \vec{\tilde{c}}_i}{\partial t} + \vec{g}_i = \vec{0}. \tag{19}$$

The structure of the mass matrix $\mathbf{M}_i$ clearly depends on the choice of the basis functions. We use an orthonormalized monomial basis that implies that the mass matrix reduces to the identity matrix $\mathbf{I}$ in standard cells. This does not hold in cells intersected by the immersed boundary, though, which we will discuss in the following section.

3.2. A discontinuous Galerkin scheme with immersed boundaries

Given an implicit representation of an immersed boundary by means of a locally smooth level set function, we partition Ω_h into the physical region

$$\mathcal{A} = \{\vec{x} \in \Omega_h : \varphi(\vec{x}) > 0\}, \tag{20}$$

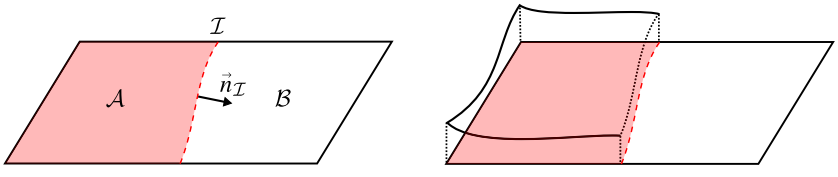
the void region

$$\mathcal{B} = \{\vec{x} \in \Omega_h : \varphi(\vec{x}) < 0\}, \tag{21}$$

and the immersed boundary

$$\mathcal{I} = \{\vec{x} \in \Omega_h : \varphi(\vec{x}) = 0\}. \tag{22}$$

A natural extension of (19) can be formulated starting from the discrete weak formulation (11) by restricting the problem domain to the physical region $\mathcal{A}$. That is, the support of the basis functions $\vec{\Phi}_i$ and the domains of integration in cell $\mathcal{K}_i$ are restricted to the sub-domains $\mathcal{A}_i = \mathcal{K}_i \cap \mathcal{A}$ and $\partial\mathcal{A}_i$ (cf. Figure 1), respectively, before performing the partial integration and inserting the numerical approximations. Taking into account that the surface $\partial\mathcal{A}_i$ consists of the edges $\{\varepsilon_{i,e}^{\mathcal{A}}\}_{e=1,\dots,E} =$



(a) Exemplary configuration in a cut-cell

(b) Polynomial approximation which is cut at $\mathcal{I}$

Figure 1. Illustration of the approximation in cut cells. The interface $\mathcal{I}$ (dashed red line) and its normal vector $\vec{n}_{\mathcal{I}}$ are implicitly defined by the level set function φ. Sub-domain $\mathcal{A}$ (light red) is associated with the fluid domain, while $\mathcal{B}$ is considered void.

DOI: 10.1002/nme

$\{\varepsilon_{i,e} \cap \overline{\mathcal{A}_i}\}_{e=1,\dots,E}$ and the boundary segment $\mathcal{I}_i = \mathcal{K}_i \cap \mathcal{I}$, it directly follows that the discrete weak formulation in the context of an IBM can be written as follows:

$$\begin{aligned} &\int_{\mathcal{A}_i} \frac{\partial \tilde{c}_i}{\partial t} \Phi_{i,j}\, dV + \sum_{e=1}^{E} \int_{\varepsilon_{i,e}^{\mathcal{A}}} g\left(\tilde{c}_i^-, \tilde{c}_i^+, \vec{n}_i\right)\, \Phi_{i,j}\, dA \\ &+ \int_{\mathcal{I}_i} g\left(\tilde{c}_i^-, \tilde{c}^{\mathrm{bc}}, \vec{n}_{\mathcal{I}}\right)\, \Phi_{i,j}\, dA - \int_{\mathcal{A}_i} \vec{g}(\tilde{c}_i) \cdot \nabla \Phi_{i,j}\, dV = 0, \end{aligned} \tag{23}$$

where $\vec{n}_{\mathcal{I}} = -\nabla\varphi / \|\nabla\varphi\|$. As a result, the semi-discrete system (19) in cells intersected by $\mathcal{I}$ is defined by

$$(\mathbf{M}_i)_{k,j} := \int_{\mathcal{A}_i} \Phi_{i,k} \Phi_{i,j}\, dV \tag{24}$$

and

$$(\vec{g}_i)_j := \sum_{e=1}^{E_i} \int_{\varepsilon_{i,e}^{\mathcal{A}}} g\left(\tilde{c}_i^-, \tilde{c}_i^+, \vec{n}_i\right)\, \Phi_{i,j}\, dA + \int_{\mathcal{I}_i} g\left(\tilde{c}_i^-, \tilde{c}^{\mathrm{bc}}, \vec{n}_{\mathcal{I}}\right)\, \Phi_{i,j}\, dA - \int_{\mathcal{A}_i} \vec{g}(\tilde{c}_i) \cdot \nabla \Phi_{i,j}\, dV. \tag{25}$$

Even though this extension is rather straightforward from a mathematical standpoint, the method strongly relies on the accurate and efficient evaluation of the integrals over $\mathcal{A}_i$ and $\mathcal{I}_i$. We will discuss this point in Section 4. Moreover, the presence of intersected cells $\mathcal{K}_i$ with extremely small volume fractions

$$\mathrm{frac}(\mathcal{K}_i) = \frac{\mathrm{meas}(\mathcal{A}_i)}{\mathrm{meas}(\mathcal{K}_i)} \tag{26}$$

inside of the physical problem domain can cause significant issues. We rectify these problems using the cell-agglomeration strategy outlined in Section 5.

3.3. Solution of the semi-discrete system

The temporal discretization of (19) with $\vec{g}_i$ defined by (25) can be carried out by virtue of standard Runga–Kutta schemes. Note that (25) simply reduces to (15) for uncut cells. Because we focus on steady-state problems within this work, we use a simple explicit Euler time discretization to drive (19) into a steady state. That is, we iterate

$$\vec{\tilde{c}}_i \leftarrow \vec{\tilde{c}}_i - \Delta t \mathbf{M}_i^{-1} \vec{g}_i(\tilde{c}) \tag{27}$$

with a pseudo time-step size Δt until a steady state is reached.

Here, we have used the modified step size restriction

$$\Delta t \leqslant \frac{c_{\mathrm{CFL}}}{2P+1} \frac{\sqrt[D]{\mathrm{meas}(\mathcal{A}_i)}}{\left\|\vec{u}\right\| + a} \tag{28}$$

with the speed of sound a and $0 < c_{\mathrm{CFL}} \leqslant 1$. Consequently, the step size is strongly influenced by the (sub-)cell $\mathcal{A}_i$ with the smallest volume. We note that this formula does not take the actual shape of the sub-cells into account, which is why we would have to choose relatively low values for the constant c_{CFL}. This step size restriction is alleviated significantly by the cell-agglomeration technique presented in Section 5, which is sufficient for the purposes of this work. However, this point requires more attention for time-dependent calculations, which will be discussed in a follow-up publication.

DOI: 10.1002/nme

4. NUMERICAL INTEGRATION

Due to the increasing popularity of the XFEM [24] and related sharp-interface approaches, the numerical integration over implicitly defined sub-domains has received significant attention in recent years. Most approaches presented in literature deliver second-order convergence rates in the presence of curved interfaces [3, 16, 18, 25–31] and are thus often supplemented by a recursive subdivision strategy in order to retain the formal convergence order of the underlying discretization scheme.

Recently, methods that aim at overcoming the need for (an excessive number of) subdivisions have gained more interest, especially in the context of the higher order XFEM [9, 10, 21, 32], the finite cell method [33], and extended/unfitted DG methods [20, 22]. The majority of these approaches are based on the idea of either constructing a higher order approximation of the interface [9, 32, 33] or computing a mapping function that improves its piecewise linear approximation [21]. A different group of methods is based on reducing the dimension of the integrals to be computed using the idea of height functions [20, 34], typically requiring a moderate number of subdivisions of intersected cells.

Yet another class of methods is based on solving the moment-fitting equations [35], which has originally been proposed for the pre-calculation of highly efficient, Gauss-like quadrature rules for arbitrary polyhedra, even though some of the underlying ideas have already been anticipated in [13]. Later, the method has been adapted to problems involving piecewise linear *moving* interfaces [36, 37]. Within this work, we make use of an extension to curved interfaces [38] whose main ideas will be summarized in Section 4.1. This extension still has to be supplemented by a node placement strategy. We will thus briefly summarize some implications of the strategy applied in [38] (cf. Section 4.2) before discussing a novel strategy that is suitable for the problems discussed in the present work (cf. Section 4.3).

4.1. Hierarchical moment-fitting

A generic strategy for the construction of quadrature rules for a given set of basis functions $\Phi_{i,j}$ in a cell $\mathcal{K}_i$ is given by the solution of the moment-fitting system

$$\underbrace{\begin{pmatrix} \Phi_{i,1}(\vec{x}_1) & \dots & \Phi_{i,1}(\vec{x}_L) \\ \vdots & \ddots & \vdots \\ \Phi_{i,M}(\vec{x}_1) & \dots & \Phi_{i,M}(\vec{x}_L) \end{pmatrix}}_{=:\mathbf{A}} \begin{pmatrix} w_1 \\ \vdots \\ w_L \end{pmatrix} = \underbrace{\begin{pmatrix} \int_{\mathcal{K}_i} \Phi_{i,1} dV \\ \vdots \\ \int_{\mathcal{K}_i} \Phi_{i,M} dV \end{pmatrix}}_{=:\vec{b}} \tag{29}$$

with nodes $\mathcal{X} = \{\vec{x}_l\}_{l=1,\dots,L}$ and weights $\mathcal{W} = \{w_l\}_{l=1,\dots,L}$ [35]. For the purposes of this work, it suffices to limit ourselves to a polynomial basis $\Phi_{i,j}$ of maximum degree Q.

The moment-fitting system is linear in the weights and strongly nonlinear in the nodes. In order to obtain an optimal quadrature rule, system (29) can be solved several times while eliminating nodes with marginal significance [39]. Even though this approach is exceedingly difficult and costly for general applications, we can greatly simplify the solution if

(i) a sufficiently accurate approximation of the right-hand side can be evaluated cheaply and
(ii) an appropriate set of quadrature nodes $\mathcal{X}$ can be predefined.

Then, (29) reduces to a linear system

$$\mathbf{A}\vec{w} = \vec{b}, \tag{30}$$

which can be solved efficiently.

Requirement (i) can be met by using a recursive strategy based on a hierarchy of integrals over lower dimensional sub-domains. That is, we trace the volume case (integral over $\mathcal{A}$) back to the surface case (integral over $\mathcal{I}$), which is further traced back to the $(D-1)$-dimensional volume case (integral over $\partial\mathcal{A}$) and so on. The base case given by integrals over lines intersected by $\mathcal{I}$

DOI: 10.1002/nme

can be evaluated by determining the roots of φ on these lines. This process has not been modified with respect to the implementation presented in [38], which is why we abstain from repeating the details here.

4.2. Original strategy with identical nodes in all cells

Regarding requirement (ii), best results have been reported using tensor-product Gauss rules defined on the full cell $\mathcal{K}_i$ [38]. A major advantage of using identical nodes in all cells is that the corresponding numerical operations can be vectorized, which significantly improves computational efficiency. This setting is illustrated in Figure 2, where the number of Gauss nodes has been chosen such that the linear system (30) is under-determined. That is, we demand that $L \geqslant \beta M$ with some safety factor $\beta > 1$, where M is the number of basis functions and thus equations. This practice guarantees that computing the least-squares solution of (30) does not introduce an additional error, where larger safety factors β improve the robustness and the conditioning of the moment-fitting system. It should be noted that the linear least-squares approach does not necessarily result in non-negative weights (cf. Figure 2).

In [22], a variant of the procedure presented in this section has been used for the simulation of incompressible multiphase flows with a sharp phase-interface. In the numerical experiments presented therein, the experimental order of convergence (EOC) matches the expected rate of $P + 1$ if a moment-fitting basis of order $Q \geqslant 2P$ is employed, which is an obvious choice for the evaluation of the weak forms. It is well known that a higher order representation of curved boundaries is required to retain the full convergence rate of DG schemes [40]. Within the present setting, this implies that the error in the evaluation of integrals over $\mathcal{A}$ and $\mathcal{I}$ must converge at the same rate as the approximation error. The steady phase interface assumed in [22] can be interpreted as an immersed boundary separating two fluid domains, which leads us to the conclusion that the applied numerical integration procedure satisfies this property, at least in the considered configurations. This is in agreement with the results presented [38], where an EOC of $Q + 1$ has been observed. On the downside, the numerical experiments in [22] revealed rare corner cases where the presence of negative weights caused a loss of positive determinacy of the mass matrix due to round-off errors. This issue had to be resolved by means of a modified Cholesky factorization.

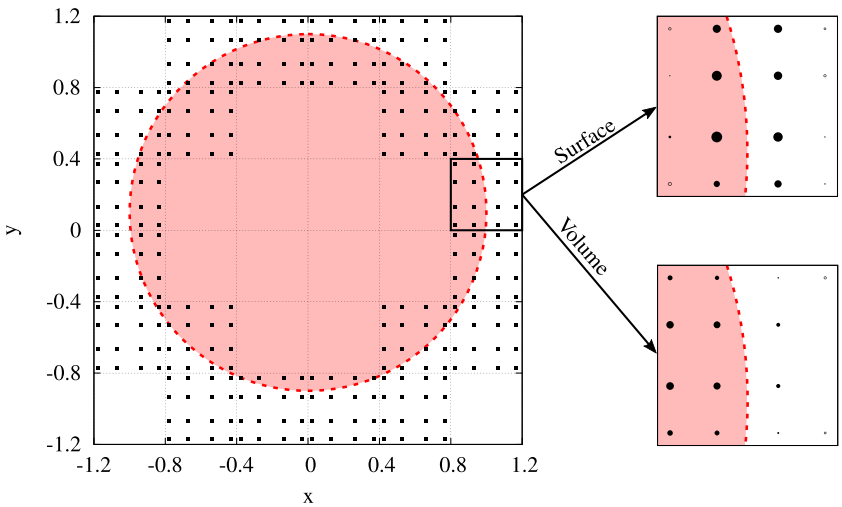


Figure 2. Quadrature nodes and weights for the example of a unit circle (light red) represented by the zero iso-contour (dashed red line) of the level set function $\varphi = x^2 + (y - 0.1)^2 - 1$. The magnitudes of the weights $\vec{w}$ depicted on the right have been obtained by means of the *original* node placement strategy [38] using moments up to order 2 and $\beta = 1.6$. Filled circles correspond to positive weights; negative weights are denoted by empty circles. Note that we have chosen the same number of nodes in the volume and in the surface case for illustrative purposes. In practice, it suffices to use *less* nodes in the volume case to obtain third-order convergence for this case.

DOI: 10.1002/nme

While the results obtained in [22] are certainly encouraging, our numerical experiments revealed that the proposed node distribution is impractical for the simulation of compressible flows. This is due to the fact that placing nodes in all parts of the cell requires a smooth extension of the problem into the void region. As we do not have control over the solution in this fictitious part of the computational domain, an extension of the density or the energy might be negative at some quadrature nodes.

We conclude that the node placement strategy illustrated in Figure 2 is *not* suitable for the present application scenario. Instead, we avoid related problems by placing all nodes in the sub-domain $\mathcal{A}$, as it has, for example, been proposed in [41] and [42].

4.3. Modified placement of the quadrature nodes

We now seek a modified node placement strategy retaining all positive features of the original quadrature scheme outlined in the previous section, except for the invariant positioning of the quadrature nodes. This is a non-trivial task because the sub-domains $\mathcal{A}_i$ may have very general, non-convex shapes that prohibit the usage of standard approaches for polytopes. Moreover, the strategy should be computationally inexpensive because our ultimate target applications feature moving domains, which implies that the quadrature rules will have to be recomputed frequently. As a result, we formulate a heuristic strategy in the following.

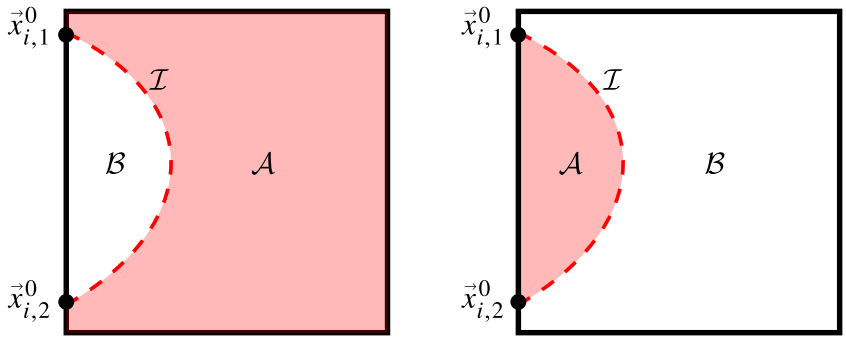
A naive strategy could be defined by using the fixed node set from the previous section and simply removing all nodes $\vec{x}_l$ that are located in $\mathcal{B}$. In order to keep the number of remaining nodes $\tilde{L}$ sufficiently large, this requires an exceedingly large factor β if small cut cells are present. As a consequence, we define the initial set of nodes on a suitable approximation of the bounding box $\mathcal{C}_i$ of $\mathcal{A}_i$. Such an approximation can easily be constructed from the vertices of $\mathcal{A}_i$. These vertices are simply given by the vertices $\{\vec{v}_{i,v}\}_{v=1,\dots,V}$ of $\mathcal{K}_i$ contained in $\mathcal{A}$, and the roots $\{\vec{x}^0_{i,r}\}_{r=1,\dots,R}$ of φ on $\partial\mathcal{K}_i$ (two-dimensional case) or $\partial(\partial\mathcal{K}_i)$ (three-dimensional case) that have been determined for the line integration mentioned in the previous section.

However, we have to ensure that the bounding box is also well defined in cases where the sign of φ is identical at all vertices and $R > 0$, that is, where all roots are located on a single edge $\varepsilon_{i,e}$ (cf. Figure 3). We can simply set $\mathcal{C}_i = \mathcal{K}_i$ if the sign is *positive* at all vertices (cf. Figure 3(a)), but the remaining case (cf. Figure 3(b)) requires a special treatment because no vertex $\{\vec{v}_{i,v}\}$ is located inside of $\mathcal{A}_i$. The actual number of roots R in a corresponding configuration can be any even number in the general case. In practical calculations, we only consider cases where φ has exactly two roots on a single edge $\varepsilon_{i,e} \in \partial\mathcal{K}_i$ ($R = 2$). We then construct an additional point by computing the mid-point

$$\vec{x}^{\,c}_i := \frac{1}{2}\left(\vec{x}^0_{i,1} + \vec{x}^0_{i,2}\right) \tag{31}$$

of the roots and moving it towards the interface via

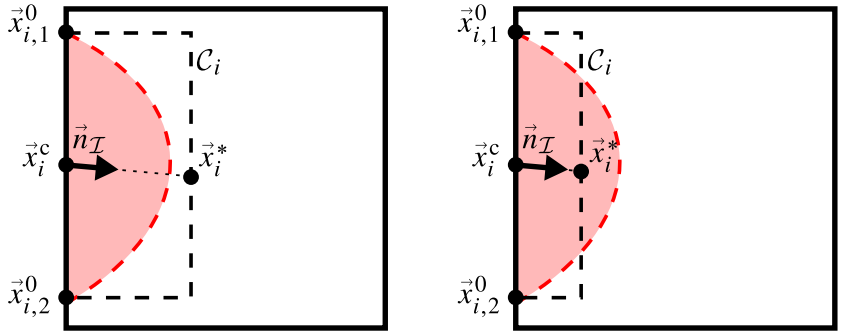
$$\vec{x}^{\,*}_i := \vec{x}^{\,c}_i + \varphi(\vec{x}^{\,c}_i)\,\vec{n}_{\mathcal{I}}. \tag{32}$$



(a) $\varphi(\vec{v}_{i,v}) > 0\ \forall v$ (b) $\varphi(\vec{v}_{i,v}) < 0\ \forall v$

Figure 3. Exemplary configurations where the sign of φ is identical at all vertices but exactly two roots have been found on the same edge ($R = 2$).

DOI: 10.1002/nme



(a) Steep level set function ($\|\nabla\varphi\| > 1$) (b) Flat level set function ($\|\nabla\varphi\| < 1$)

Figure 4. Potential bounding boxes $\mathcal{C}_i$ (dashed black lines) for the configuration sketched in Figure 3(b), depending on the local value of $\|\nabla\varphi\|$.

We emphasize that this approach can easily be extended to configurations with $R > 2$, for example, by computing the mid-point of each pair of roots. Note that pairs of roots can trivially be identified in the two-dimensional setting because all R roots are located on a single edge. We have still chosen to exclude this case because $R > 2$ implies an excessively coarse resolution near the immersed boundary $\mathcal{I}$ where accuracy requirements are typically highest. As a consequence, we *intentionally* reject grids where $R > 2$ for the purposes of this paper.

It should also be noted that we do not require φ to satisfy the signed distance property (cf. Section 6), which is why $\vec{x}^*_i$ does not exactly reside on $\mathcal{I}$ in most applications. An exemplary configuration where $\|\nabla\varphi\| > 1$ is sketched in Figure 4(a). Here, the bounding box $\mathcal{C}_i$ covers $\mathcal{A}_i$ completely, but this property is lost if $\|\nabla\varphi\| < 1$ holds (cf. Figure 4(b)). Obviously, this problem could be attenuated by moving $\vec{x}^*_i$ closer to $\mathcal{I}$ via an iterative procedure. However, we did not observe any negative effects in either case, which can be understood by recalling that the moment-fitting system (29) is still under-determined if a sufficiently large safety factor β is used. The location of the quadrature nodes only changes the properties of $\mathbf{A}$, whereas the right-hand $\vec{b}$ remains unchanged. As a result, it suffices to verify that $\mathbf{A}$ is not ill conditioned, for example by checking that the numerical residual $\left\|\mathbf{A}\vec{w} - \vec{b}\right\|$ is close to machine accuracy.

Finally, the bounding box $\mathcal{C}_i$ is constructed from $(\{\vec{v}_{i,v}\} \cap \overline{\mathcal{A}}) \cup \{\vec{x}^0_{i,r}\}$ where applicable and from $\{\vec{x}^0_{i,1}, \vec{x}^0_{i,2}, \vec{x}^*_i\}$ in the remaining cells. Within $\mathcal{C}_i$, we seed equidistant nodes with at least $L \geqslant \beta M$ nodes, before eliminating all nodes that reside in $\mathcal{B}$. This simple heuristic strategy greatly improves quadrature performance compared with the naive seeding discussed previously, and our numerical experiments presented in Section 6 indicate that the strategy is sufficient to retain the optimal convergence rate of the DG scheme in a wide variety of configurations using $\beta = 5$.

An exemplary output of the modified algorithm is displayed in Figure 5. A positive side effect of the modified node placement strategy is the drastic reduction of negative weights in the volume case. In effect, we did not encounter any problems with the numerically indefinite mass matrices reported in [22] (cf. previous section) in any of our numerical experiments; that is, the standard Cholesky algorithm could be applied to invert the mass matrices.

5. CELL-AGGLOMERATION

Up to this point, the discretization outlined in Section 3.2 admits arbitrarily small cut cells with arbitrary shapes. Obviously, this renders the method impractical because of the extreme time-step restriction in cells with small volume fractions (cf. Figure 6(a)). Moreover, our numerical experiments suggest that the presence of *sliver* elements with large aspect ratios have a strongly negative effect for $P \geqslant 2$. In Figure 6(b), we give an example for a corresponding cell $\mathcal{K}$ with a volume fraction of about 20%. While we did not encounter any problems for low-order simulations ($P \leqslant 1$), using a higher order basis $\vec{\Phi}$ defined on $\mathcal{K}$ resulted in strongly ill-conditioned mass matrices.

DOI: 10.1002/nme

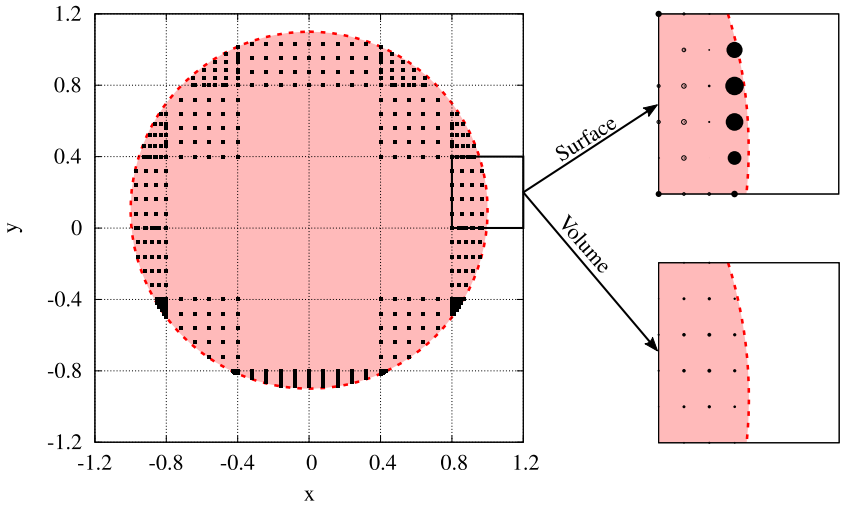
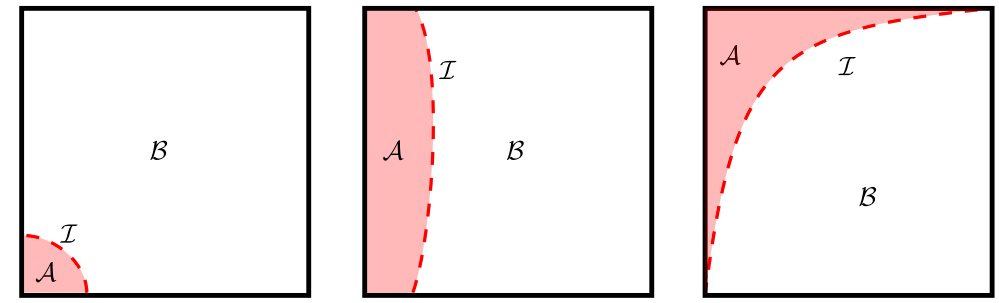


Figure 5. Quadrature nodes and weights for the example of a unit circle (light red) represented by the zero iso-contour (dashed red line) of the level set function $\varphi = x^2 + (y - 0.1)^2 - 1$. The magnitudes of the weights $\vec{w}$ depicted on the right have been obtained by means of the *modified* node placement strategy using moments up to order 2 and $\beta = 5$. Filled circles correspond to positive weights; negative weights are denoted by empty circles. Note that we have chosen the same number of nodes in the volume and in the surface case for illustrative purposes. In practice, it suffices to use *less* nodes in the volume case to obtain third-order convergence for this case.



(a) Sub-cell with a small volume fraction

(b) Sliver sub-cell

(c) Strongly curved sub-cell

Figure 6. Configurations where cell-agglomeration is required for higher order simulations.

This finding can be interpreted as a loss of linear independence of the basis functions within $\mathcal{A}$ that has to be addressed.

A simple remedy for this problem is the definition of the basis on the bounding box of $\mathcal{A}$, which has for example been proposed in [3] for $P \leqslant 2$. Unfortunately, this approach still leads to problematic situations if the immersed boundary is strongly curved. In Figure 6(c), we give an example for a cell with approximately the same volume fraction as in the second example (Figure 6(b)), but where the bounding box of $\mathcal{A}$ coincides with $\partial\mathcal{K}$. As a result, rescaling the basis to the bounding box of $\mathcal{A}$ is insufficient when higher approximation orders ($P \geqslant 3$) are considered.

In order to overcome these issues, we employ the cell-agglomeration strategy proposed in [22], which we will outline briefly in the following. We will then present a novel reformulation of this strategy, which is particularly useful for the supplementation of existing explicit DG schemes (cf. Section 5.2). Finally, we will discuss the effect of the cell-agglomeration procedure on the discretization in Section 5.3.

DOI: 10.1002/nme

5.1. Agglomeration strategy

In the first step of the algorithm proposed in [22], we determine the list of agglomeration *source* cells $\{\mathcal{K}_s^{\text{src}}\}_{s=1,\dots,S} = \{\mathcal{K}_i : \text{frac}(\mathcal{K}_i) \leqslant \delta\}$ based on the volume fraction (26) with a user-defined threshold $0 \leqslant \delta < 1$. Next, we determine the agglomeration *target* cell $\mathcal{K}_s^{\text{tar}}$ for each source cell $\mathcal{K}_s^{\text{src}}$ by finding the edge neighbor $\{\mathcal{N}_{s,e}\}_{e=1,\dots,E} = \{\mathcal{K}_i : \overline{\mathcal{K}_i} \cap \overline{\mathcal{K}_s} = \varepsilon_{s,e}\}$ with the largest volume fraction; that is,

$$\mathcal{K}_s^{\text{tar}} = \arg\max_{\mathcal{N}_{s,e}} \text{frac}(\mathcal{N}_{s,e}). \tag{33}$$

A major advantage of DG methods is the weak coupling of neighboring cells via fluxes. As a result, each basis $\vec{\Phi}_i$ can be chosen independently, which simply allows us to extend the basis of the target cell $\mathcal{K}_s^{\text{tar}}$ into the agglomeration source cell $\mathcal{K}_s^{\text{src}}$. The source cell is then formally eliminated from the discretization, thus defining the agglomerated mesh $\{\mathcal{K}_i^{\text{agg}}\}_{i=1,\dots,N}$ via

$$\mathcal{K}_i^{\text{agg}} = \begin{cases} \emptyset & \mathcal{K}_i \in \{\mathcal{K}_s^{\text{src}}\} \\ \mathcal{K}_i \cup \{\mathcal{K}_s^{\text{src}} : \mathcal{K}_s^{\text{tar}} = \mathcal{K}_i\} & \text{otherwise.} \end{cases} \tag{34}$$

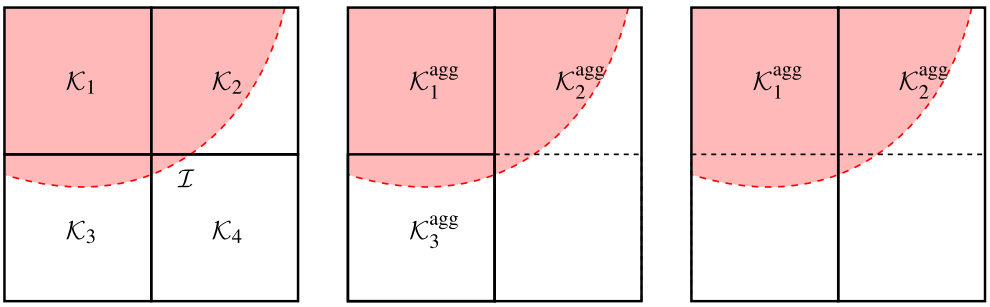
This does however not reflect our actual implementation (cf. Section 5.2), which exploits the locality of the involved operations and hence only requires few additional *cell-local* matrix–vector products per time-step that hardly affect the overall performance.

The cell-agglomeration process is illustrated in Figure 7 using the example depicted in Figure 7(a) that initially consists of four cells. Depending on the selected threshold δ, the resulting mesh consists of three (Figure 7(b)) or two cells (Figure 7(c)). Here, it is worth noting that cell $\mathcal{K}_4$ is not agglomerated to cell $\mathcal{K}_1$ because they do not share a common edge. However, further increasing δ would lead to a situation where $\mathcal{K}_2^{\text{agg}}$ from Figure 7(c) would in turn be agglomerated to $\mathcal{K}_1^{\text{agg}}$, which would indeed link cell $\mathcal{K}_4$ to cell $\mathcal{K}_1$. While this case can easily be handled via a recursive strategy, we did not encounter a relevant configuration during our numerical experiments presented in Section 6, which is why we abstain from discussing the details here.

5.2. Implementation

For the discussion of the implementation, we limit ourselves to a single agglomeration pair consisting of the source cell $\mathcal{K}^{\text{src}}$ and the target cell $\mathcal{K}^{\text{tar}}$ that are merged to form $\mathcal{K}^{\text{agg}}$. Let $\vec{\Phi}^{\text{tar}} = (\Phi_1^{\text{tar}}, \dots, \Phi_M^{\text{tar}})$ and $\vec{\Phi}^{\text{src}} = (\Phi_1^{\text{src}}, \dots, \Phi_M^{\text{src}})$ be the row vectors of basis functions of $\mathcal{K}^{\text{tar}}$ and $\mathcal{K}^{\text{src}}$, respectively. Note that the individual basis functions are non-zero in the corresponding cells only. We then construct a coupling matrix $\mathbf{Q} \subset \mathbb{R}^{M,M}$ such that $\vec{\Phi}^{\text{src}}\mathbf{Q}$ is the smooth extension of $\vec{\Phi}^{\text{tar}}$ into $\mathcal{K}^{\text{tar}}$. The basis of $\mathcal{K}^{\text{agg}}$ is hence defined as follows:

$$\vec{\Phi}^{\text{agg}} = \vec{\Phi}^{\text{tar}} + \vec{\Phi}^{\text{src}}\mathbf{Q}. \tag{35}$$



(a) Initial mesh (b) Agglomerated mesh, δ small (c) Agglomerated mesh, δ large

Figure 7. Illustration of the cut cell-agglomeration strategy. For small values of δ (middle), only cell $\mathcal{K}_4$ (*source*) will be agglomerated to the *direct* neighbor with the largest volume fraction, namely, cell $\mathcal{K}_2$ (*target*). If δ is increased (right), $\mathcal{K}_3$ will additionally be agglomerated to cell $\mathcal{K}_1$.

DOI: 10.1002/nme

In order to obtain $\mathbf{Q}$, we project the smooth extension $\underline{\vec{\Phi}}^{\text{tar}} = (\underline{\Phi}_1^{\text{tar}}, \ldots, \underline{\Phi}_M^{\text{tar}})$ of the basis $\vec{\Phi}^{\text{tar}}$ onto $\mathcal{K}^{\text{src}}$. That is, we compute

$$\mathbf{Q} = \begin{pmatrix} \int\limits_{\mathcal{K}^{\text{src}}} \underline{\Phi}_1^{\text{tar}} \Phi_1^{\text{src}} dV & \ldots & \int\limits_{\mathcal{K}^{\text{src}}} \underline{\Phi}_M^{\text{tar}} \Phi_1^{\text{src}} dV \\ \vdots & & \vdots \\ \int\limits_{\mathcal{K}^{\text{src}}} \underline{\Phi}_1^{\text{tar}} \Phi_M^{\text{src}} dV & \ldots & \int\limits_{\mathcal{K}^{\text{src}}} \underline{\Phi}_M^{\text{tar}} \Phi_M^{\text{src}} dV \end{pmatrix} \tag{36}$$

once at the beginning of a simulation, from which it directly follows that

$$\underline{\vec{\Phi}}^{\text{tar}} = \vec{\Phi}^{\text{src}} \mathbf{Q}. \tag{37}$$

The benefits of definition (35) are two-fold. First, it greatly simplifies the extension of an existing implementation of a DG scheme because all operators defined in Sections 3.1 and 3.2 can be reused without changing the underlying data structures. We will outline this process in the following. Second, the introduction of $\mathbf{Q}$ greatly simplifies the handling of dynamic (de-)agglomeration of cells in transient calculations with moving boundaries. The second point is beyond the scope of this work, which is why we will not discuss the details here.

In the setup phase of the computation, we need to compute the mass matrix $\mathbf{M}^{\text{agg}}$ as well as the initial condition for the agglomerated cells, whereas we need to compute the agglomerated fluxes in every time-step. It is easy to verify that these quantities can directly be computed from the respective quantities defined on the non-agglomerated mesh (cf. Appendix A).

The mass matrix of the agglomerated cell follows from

$$\mathbf{M}^{\text{agg}} = \mathbf{M}^{\text{tar}} + \mathbf{Q}^{\text{T}} \mathbf{M}^{\text{src}} \mathbf{Q}, \tag{38}$$

which allows us to compute the projection of the initial condition via

$$\vec{\tilde{c}}^{\text{agg}} = (\mathbf{M}^{\text{agg}})^{-1} \left(\mathbf{M}^{\text{tar}} \vec{\tilde{c}}^{\text{tar}} + \mathbf{Q}^{\text{T}} \mathbf{M}^{\text{src}} \vec{\tilde{c}}^{\text{src}} \right). \tag{39}$$

After this projection, the solution stays continuous within $\mathcal{K}^{\text{agg}}$ for all times. As a result, injecting the agglomerated solution $\vec{\tilde{c}}^{\text{agg}}$ into the non-agglomerated space via

$$\vec{\tilde{c}}^{\text{tar}} = \vec{\tilde{c}}^{\text{agg}} \tag{40}$$

and

$$\vec{\tilde{c}}^{\text{src}} = \mathbf{Q} \vec{\tilde{c}}^{\text{agg}} \tag{41}$$

leads to an equivalent problem formulation *on the non-agglomerated mesh*. The updated solution is then computed by means of

$$\vec{\tilde{c}}^{\text{tar}} \leftarrow \vec{\tilde{c}}^{\text{tar}} - \Delta t \, (\mathbf{M}^{\text{agg}})^{-1} \vec{g}^{\text{agg}}, \tag{42}$$

$$\vec{\tilde{c}}^{\text{src}} \leftarrow \mathbf{Q} \vec{\tilde{c}}^{\text{tar}}, \tag{43}$$

where the flux vector $\vec{g}^{\text{agg}}$ can be computed from the non-agglomerated fluxes $\vec{g}^{\text{tar}}$ for $\mathcal{K}_s^{\text{tar}}$ and $\vec{g}^{\text{src}}$ for $\mathcal{K}_s^{\text{src}}$ by means of

$$\vec{g}^{\text{agg}} = \vec{g}^{\text{tar}} + \mathbf{Q}^{\text{T}} \vec{g}^{\text{src}}. \tag{44}$$

In summary, this new formulation of the cell-agglomeration algorithm allows for a simple extension of existing explicit DG schemes because the modified variable update only requires two additional, cell-local matrix–vector products (Equations (43) and (44)) per time-step. Optimized evaluation strategies for the discrete operators can thus be reused without modifying the underlying data structures.

DOI: 10.1002/nme

5.3. Choice of the agglomeration threshold

A reasonable choice for the agglomeration threshold δ introduced in Section 5.1 is not directly obvious because of its ambivalent influence. On the one hand, larger values improve the conditioning of mass matrices (cf. Section 6.1) and allow for larger explicit time-steps (cf. Equation (28)). On the other hand, this is accompanied by larger spatial discretization errors near the immersed boundary.

Reasons for particular choices are rarely discussed in literature. In [22], for example, the author demonstrates that $\delta = 0.1$ is sufficient to attain reasonably conditioned system matrices if suitable preconditioning is applied. In contrast, the authors of [3] propose using $\delta = 0.5$. According to estimate (28), these choices correspond to time-steps that are approximately 68%, respectively, 29% smaller than the maximum stable time-step on a uniform background mesh. Our numerical experiments (cf. Section 6.1) suggest that, at least for the setting discussed in this work, there is already a significant influence on solution accuracy for $\delta \geqslant 0.4$. This effect is most prominent on coarse grids with higher polynomial degrees ($P \geqslant 2$), which is obviously problematic because being able to use coarse grids is one of the main benefits of higher order methods. Moreover, larger values of δ tend to increase the influence of small movements of the immersed boundary on a given background mesh. This is due to the fact that slight movements of the interface can induce strong changes in the agglomeration pattern and thus in the effective size of boundary cells. We assess this issue in the numerical experiments presented in Section 6.2.

On the other hand, agglomeration has a stronger impact on mass matrix conditioning and hence the solution for higher approximation orders P (cf. Section 6.1). For $P = 4$ (cf. Section 6.2), for example, we observed a significant impact on the rate of convergence for $\delta \leqslant 0.2$. This finding can be traced back to numerical round-off during mass matrix inversion. The said issue does however not seem to be relevant for $P \leqslant 3$.

All in all, we draw the conclusion that using $\delta = 0.3$ is a reasonable compromise for the two-dimensional setting discussed in this work. We emphasize that this choice is *not* optimal for each individual calculation but resulted in the best overall performance when considering a range of approximation degrees P. In practice, low values for δ (e.g., $0.1 \leqslant \delta \leqslant 0.3$) are sufficient for $P \leqslant 2$, whereas $\delta \geqslant 0.3$ is desirable for $P \geqslant 3$. The results presented in Section 6 have thus been obtained with $\delta = 0.3$ if not stated otherwise. Note that this allows us to use explicit time-steps that have approximately 45% the size of the maximum stable time-step on the corresponding background mesh.

6. NUMERICAL RESULTS

In the following, we present numerical results for three two-dimensional test cases for the Euler equations, namely, the flow around a cylinder, the flow over a Gaussian bump, and the flow around an NACA0012 airfoil. The Euler equations are discretized using the HLLC flux in the form presented in [43].

We additionally consider the viscous flow around this airfoil to study the flexibility of the presented approach. For the viscous terms, we use the SIPG discretization presented in [23, Section 3], which contains two parameters: the local length scale h_e of some edge ε_e and the global scaling factor C_{IP}. In [22, Section 3.3.2], the author provides theoretical and numerical evidence indicating that the length scale $h_e = |\mathcal{A}|/|\partial\mathcal{A}|$ in combination with the cell-agglomeration strategy (cf. Section 5.1) and a penalty scaling factor of $C_{\mathrm{IP}} = 5.0$ ensures coercivity. Here, the factor C_{IP} accounts for the potentially irregular shape of cut cells [22, 44, 45]. Note that $|\mathcal{A}|$ and $|\partial\mathcal{A}|$ are available from the construction of the cut cell quadrature (cf. Section 4.3) and can hence be evaluated with no additional computational costs.

We have chosen $Q = 2P$ and $\beta = 5$ to determine the initial number of quadrature nodes (cf. Section 4) in all cases. This conservative choice ensures that the EOC is not affected by the numerical integration, especially on the finest grids for $P = 4$. Our numerical experiments indicate that using $\beta = 3$ would be sufficient for practically relevant configurations, thus lowering the computational effort.

DOI: 10.1002/nme

In case of the Euler equations, we use standard adiabatic slip wall conditions at solid boundaries (for example, see [46]) and Dirichlet conditions in the free-stream. Furthermore, we consider adiabatic walls in the case of the Navier–Stokes equations that have been implemented as presented in [23].

6.1. *Inviscid flow around a cylinder*

We first simulate the inviscid flow around a circular cylinder at Mach 0.2. The cylinder has a radius of 1 and is defined via the level set function

$$\varphi(\vec{x}) = x^2 + y^2 - 1, \tag{45}$$

which can exactly be represented by a second-order DG basis. For the spatial discretization, we use a sequence of stretched Cartesian meshes with 32×32, 64×64, 128×128, and 256×256 cells. Polynomial degrees P are varied from 0 to 3.

Figure 8 depicts the upper half of the problem domain Ω_h and the coarsest corresponding mesh. A zoom into the region around the cylinder for the two coarsest meshes is given in Figure 9. We emphasize that the cylinder is approximated by very few cells in both cases. Moreover, both meshes feature problematic cells with volume fractions of less than 5% and strong curvature, which also holds true for the finer meshes (not shown).

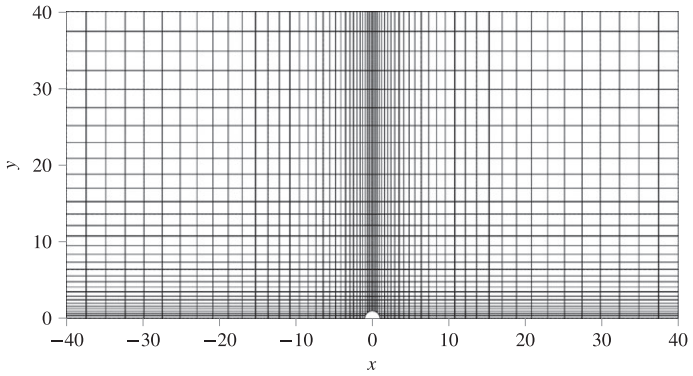
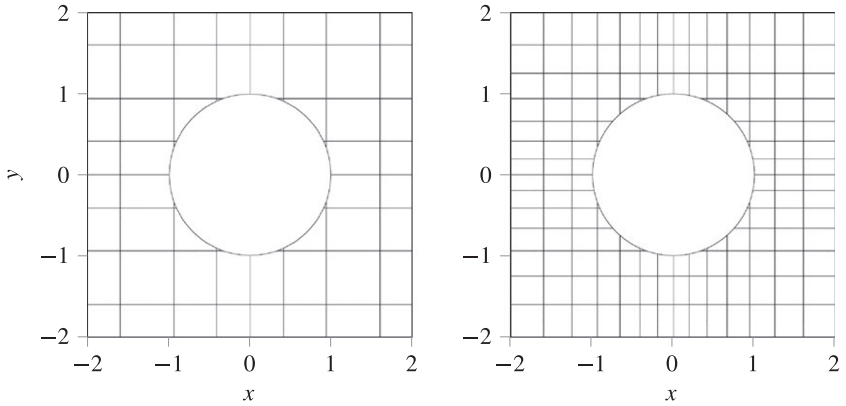


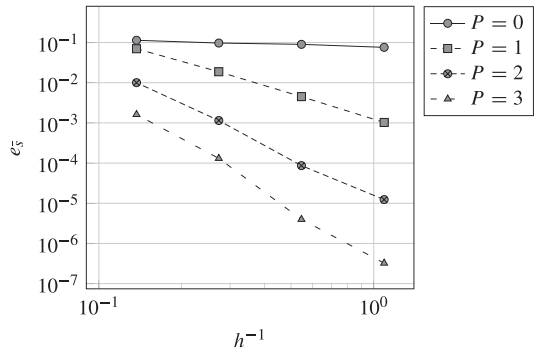
Figure 8. Upper half of the coarsest mesh for the flow around a cylinder consisting of 32×32 rectangular cells. Note that here and in the following, immersed structures have been added during post-processing for illustrative purposes.



(a) 32×32 cells (b) 64×64 cells

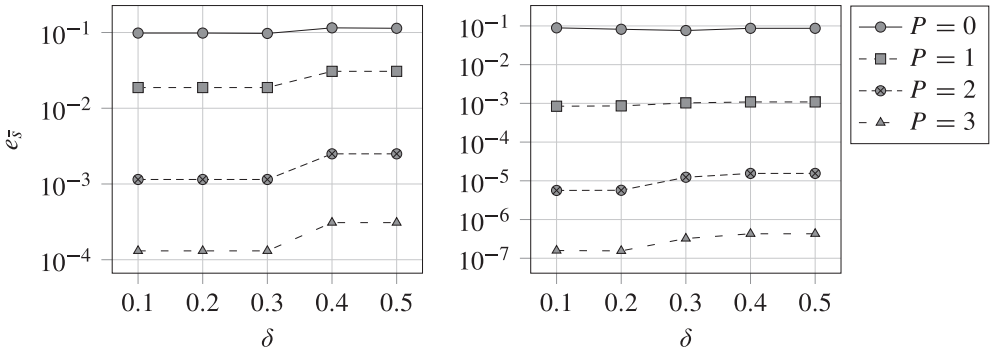
Figure 9. Zoom into the region around the cylinder on the two coarsest grids. The smallest cut cells have a volume fraction of about 5% in all configurations.

DOI: 10.1002/nme



P	EOC
0	0.19
1	2.04
2	3.28
3	4.20

Figure 10. Results of the h-convergence study for the flow around cylinder using an immersed boundary method.



(a) 64×64 cells (b) 256×256 cells

Figure 11. Entropy error as a function of the agglomeration threshold δ for the inviscid flow around a cylinder on two selected grids.

In the first step, we study the influence of the agglomeration threshold δ on the entropy error

$$e_{\bar{s}} = \sqrt{\int_{\Omega} (\Delta s)^2 dV}, \tag{46}$$

where $\Delta s = p/\rho^{\gamma} - \bar{s}_{\infty}$ is defined with respect to the free-stream entropy $\bar{s}_{\infty} = p_{\infty}/\rho_{\infty}^{\gamma} = 1$. To this end, we perform h-convergence studies for $\delta \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, where $e_{\bar{s}}$ is evaluated using the quadrature techniques presented in Section 4.

Results for $\delta = 0.3$ are displayed in Figure 10. We observe an EOC of approximately $P + 1$ in all cases except for $P = 0$, where the spatial resolution is much too coarse. We abstain from enumerating the results for all values of δ because they are qualitatively similar, even though the individual error levels vary. This finding is visualized in Figure 11 using the example of grids consisting of 64×64 and 256×256 cells, respectively. We observe that the error starts increasing significantly for $\delta \geqslant 0.3$. This trend is most prominent on coarse grids for higher polynomial degrees. On the other hand, studying the maximum of the condition numbers of all cell-local mass matrices $\mathbf{M}_i$ as a function of the agglomeration threshold (Figure 12) reveals that low values of δ ($0.1 \leqslant \delta \leqslant 0.3$) are sufficient for retaining a reasonably conditioned problem for most cases within the present setting. Still, round-off errors introduced during mass matrix inversion can be decreased significantly by increasing δ (i.e., $0.3 \leqslant \delta \leqslant 0.5$), especially on the finest grid and for $P \geqslant 3$. In combination, these observations motivate the compromise $\delta = 0.3$, which has been discussed in Section 5.3.

DOI: 10.1002/nme

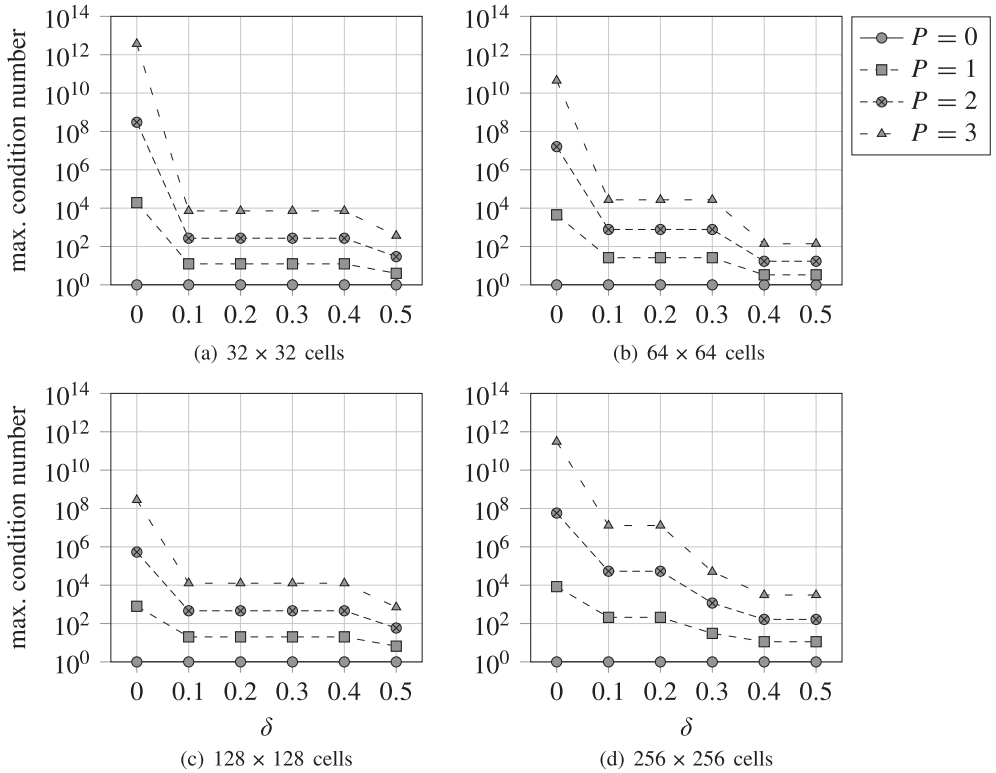


Figure 12. Study of the maximum of the condition numbers of the mass matrices $\mathbf{M}_i$ over all cells as a function of the agglomeration threshold δ using the example of the inviscid flow around a cylinder.

In a second step, we visually inspect the accuracy per DOF of the scheme. Figure 13 shows the pressure distribution and pressure contours in the vicinity of the cylinder, which should be smooth and perfectly symmetric in the ideal case. The first-order solution on the second finest grid (cf. Figure 13(a)) still exhibits jumps in the pressure contours and, especially in close vicinity of the cylinder, an asymmetric pressure field. In contrast, the second-order solution on a coarser grid (cf. Figure 13(b)) leads to smooth pressure contours that are almost perfectly symmetric, even though 50% less DOF are used. The same trend can be observed for the third-order solution on the coarsest grid (cf. Figure 13(c)) with approximately 80% less DOF than in the first-order case. While the pressure contours still feature minor discontinuities that stem from the coarse grid resolution, the accuracy per DOF is excellent.

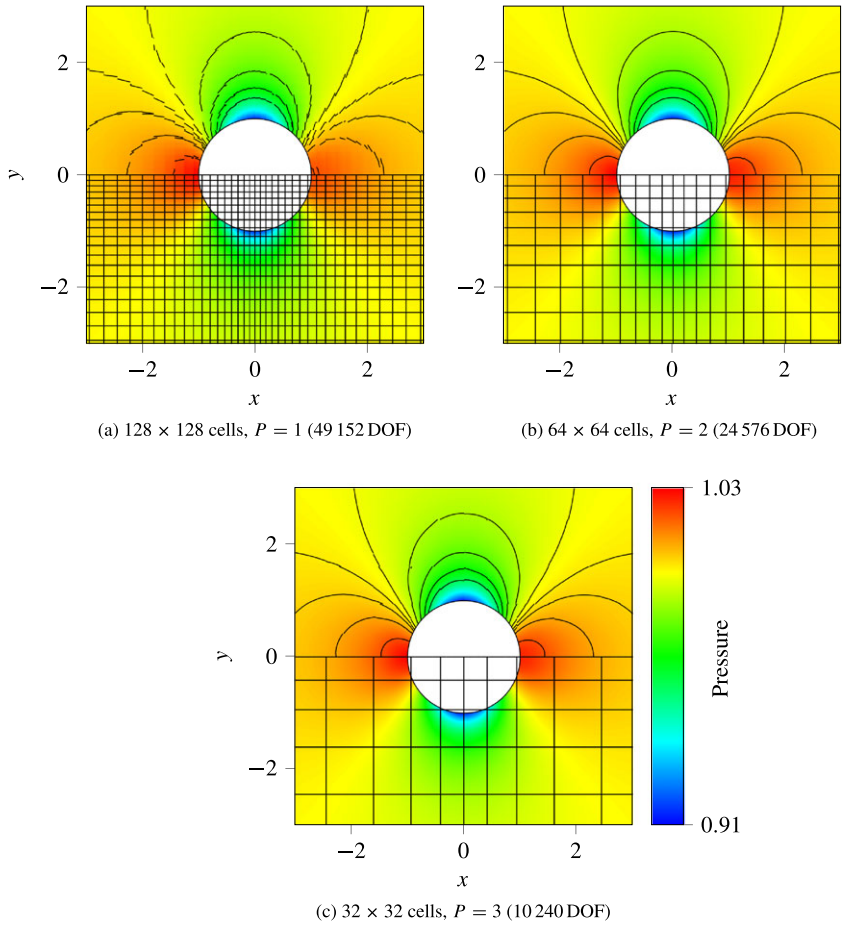
6.2. Inviscid flow over a Gaussian bump

In this test case, the flow over a smooth bump is investigated in order to study the robustness of the scheme with respect to movements of the level set. The entropy $\bar{s}$ is constant in the flow field, and the L_2-norm of the entropy error (46) is used as an error measure because the analytical solution is unknown. The smooth bump is given by the level set function

$$\varphi(\vec{x}) = (y - \epsilon_2) - 0.01 + \frac{1}{\sqrt{2\pi}} e^{-0.5(x-\epsilon_1)^2}, \tag{47}$$

where ϵ_1 and ϵ_2 are parameters that control the level set movement in x-direction and y-direction, respectively. Due to the fact that a high-order boundary representation is needed for accurate results, the level set is represented by an eighth-order DG approximation; that is, $P_{LS} = 8$. The computational domain $[-20, 20] \times [0, 20]$ is discretized using a series of equidistant Cartesian grids. Slip wall boundary conditions are applied at the smooth bump, and uniform free-stream conditions with Mach 0.5 are applied at all other boundaries.

DOI: 10.1002/nme



(a) 128 × 128 cells, $P = 1$ (49 152 DOF)

(b) 64 × 64 cells, $P = 2$ (24 576 DOF)

(c) 32 × 32 cells, $P = 3$ (10 240 DOF)

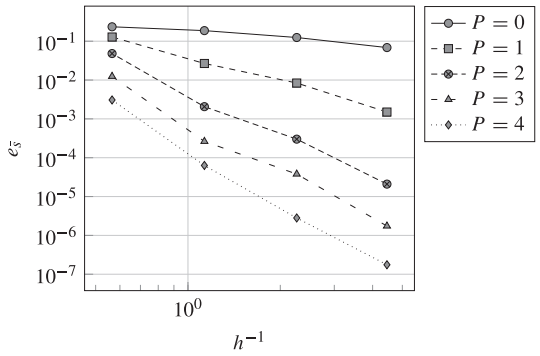
Figure 13. Comparison of pressure distribution and pressure iso-contours for the inviscid flow around a cylinder at different resolutions.

In Figure 14, we present a study of the entropy error under a uniform grid refinement for $P = 0, \ldots, 4$, starting from 32×16 cells. The level set function has not been moved in this base case; that is, $\epsilon_1 = \epsilon_2 = 0$. For $P \geqslant 1$, we observe that the error converges towards 0 with the expected rate of $P + 1$. For $P = 4$, the EOC is slightly decreased on the finer meshes. This is due to the fact that the condition number of *unmerged* cut cells grows with increasing P. In the finest case, the magnitude of the condition numbers is between 10^8 and 10^9. Calculating the inverse of the corresponding mass matrices using double-precision arithmetic hence introduces additional round-off errors in the range of 10^{-8} to 10^{-7}. This problem can easily be alleviated by further increasing the agglomeration threshold δ, however at the expense of an increase of the error level (cf. Section 6.1).

In Figure 15, we compare the pressure distribution and contours on a coarse mesh (64×32 cells) with $P = 4$ to a mesh that has been refined twice (256×128) with $P = 1$. Both simulations lead to nearly symmetric pressure profiles. However, the low-order solution (left) still contains visible jumps in the vicinity of the bump, while the high-order simulation (right) leads to smooth pressure contours using 70% less DOF.

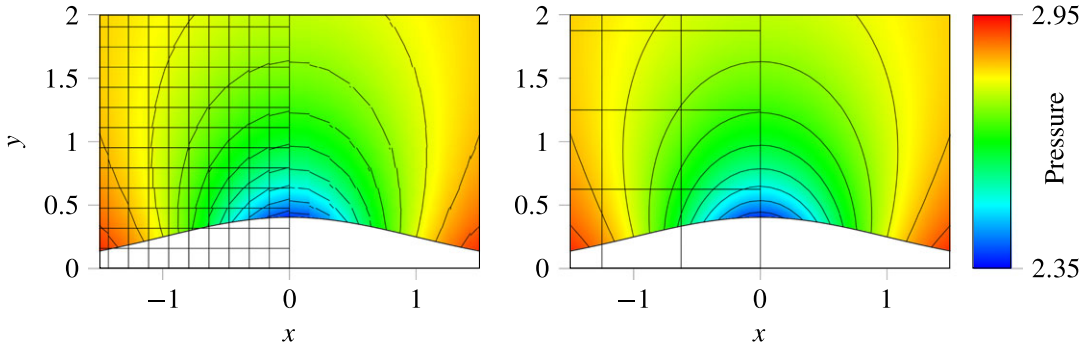
We now investigate the influence of small perturbations of the level set position for the purpose of assessing the robustness of our scheme. Therefore, we compare the deviation of the error from the base case with $P = 4$ for small changes of ϵ_1 and ϵ_2 with a magnitude of 1% or less. Table I indicates that small perturbations of the level set in y-direction only have a small influence on the solution. The maximum deviation is below 4%. Perturbations in x-direction have even less influence (below 0.5%), which is why the values are not given here. Table II additionally shows

DOI: 10.1002/nme



P	EOC
0	0.46
1	2.11
2	3.65
3	4.15
4	4.73

Figure 14. Results of the h-convergence study for the flow over a smooth bump with $\epsilon_1 = \epsilon_2 = 0$.



(a) 256 × 128 cells, $P = 1$ (98 304 DOF) (b) 64 × 32 cells, $P = 4$ (30 720 DOF)

Figure 15. Comparison of pressure distribution and contours for the inviscid flow over a smooth bump.

Table I. Comparison of the entropy error $e_{\bar{s}}$ and deviation from the base case for different values of ϵ_2, from coarse grid (first row) to fine grid (fourth row).

$\epsilon_2 = -0.01$		$\epsilon_2 = -0.005$		$\epsilon_2 = 0$	$\epsilon_2 = 0.005$		$\epsilon_2 = 0.01$	
$e_{\bar{s}}$	%	$e_{\bar{s}}$	%	$e_{\bar{s}}$	$e_{\bar{s}}$	%	$e_{\bar{s}}$	%
$3.07 \cdot 10^{-03}$	1.32	$3.01 \cdot 10^{-03}$	0.67	$3.03 \cdot 10^{-03}$	$3.05 \cdot 10^{-03}$	0.65	$2.99 \cdot 10^{-03}$	1.32
$6.32 \cdot 10^{-05}$	0.46	$6.31 \cdot 10^{-05}$	0.23	$6.30 \cdot 10^{-05}$	$6.28 \cdot 10^{-05}$	0.23	$6.27 \cdot 10^{-05}$	0.47
$2.81 \cdot 10^{-06}$	3.87	$2.98 \cdot 10^{-06}$	2.05	$2.92 \cdot 10^{-06}$	$2.87 \cdot 10^{-06}$	1.91	$2.82 \cdot 10^{-06}$	3.53
$1.67 \cdot 10^{-07}$	1.39	$1.67 \cdot 10^{-07}$	1.43	$1.65 \cdot 10^{-07}$	$1.66 \cdot 10^{-07}$	0.69	$1.66 \cdot 10^{-07}$	0.81

Table II. EOC for different values of ϵ_2.

ϵ_2	−0.01	−0.005	0	0.005	0.01
EOC	4.73	4.71	4.73	4.73	4.72

the corresponding convergence rates. Evidently, the EOC is hardly affected by the positioning of the level set, despite the fact that the agglomeration pattern changes depending on ϵ_1 and ϵ_2. We conclude that small movements of the level set have no significant influence on the solution and the convergence properties of the presented schemes for $\delta = 0.3$.

6.3. Inviscid flow over a NACA0012 profile

In this example, we consider the subsonic inviscid flow around a NACA0012 airfoil at an angle of attack of $\alpha = 2°$. The airfoil surface is specified via

$$y = \pm 0.6 \left(0.2969\sqrt{x} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1306x^4\right). \tag{48}$$

DOI: 10.1002/nme

To obtain a polynomial level set function, Equation (48) is reordered and squared, which yields the two-level set functions

$$\varphi(\vec{x}) = \left(\pm y + 0.6\left(0.126x + 0.3516x^2 - 0.2843x^3 + 0.1306x^4\right)\right)^2 - 0.03173x \qquad (49)$$

for the upper (+) and lower (−) parts. We set the approximation order of the level set field to $P_{LS} = 8$.

The outer boundary of the computational domain is 20 chord length away from the airfoil; that is, the computational box is set to $[-20, 20] \times [-20, 20]$. We use a sequence of stretched structured grids with 32×32, 64×64, and 128×128 cells, respectively. Figure 16 visualizes the refined region around the airfoil, where the resolution is highest near the tip and the tail.

Free-stream conditions are set to Mach 0.5 and $\bar{s}_\infty = 1$. Traditionally, the angle of attack α is specified by adjusting the free-stream conditions, which we will refer to as the *non-rotated* case in the following. To demonstrate the geometrical flexibility of the IBM scheme, we compare this approach to results obtained by rotating the level set function instead, henceforth referred to as the *rotated* case. We note, however, that we ensure that the trailing edge is always located at a cell

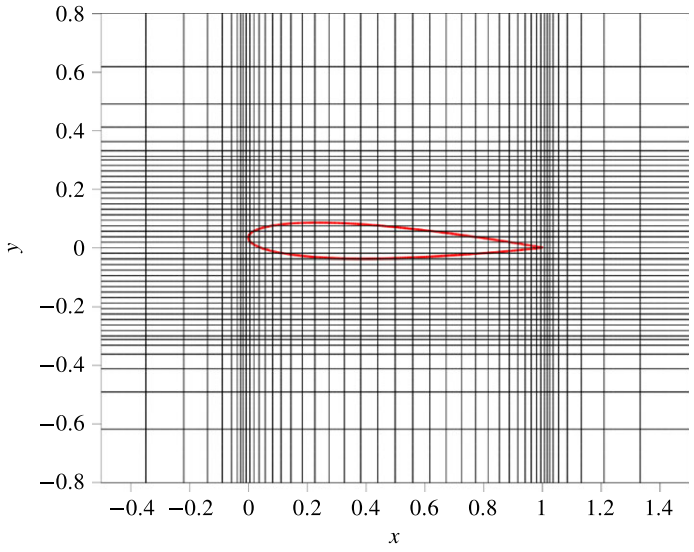
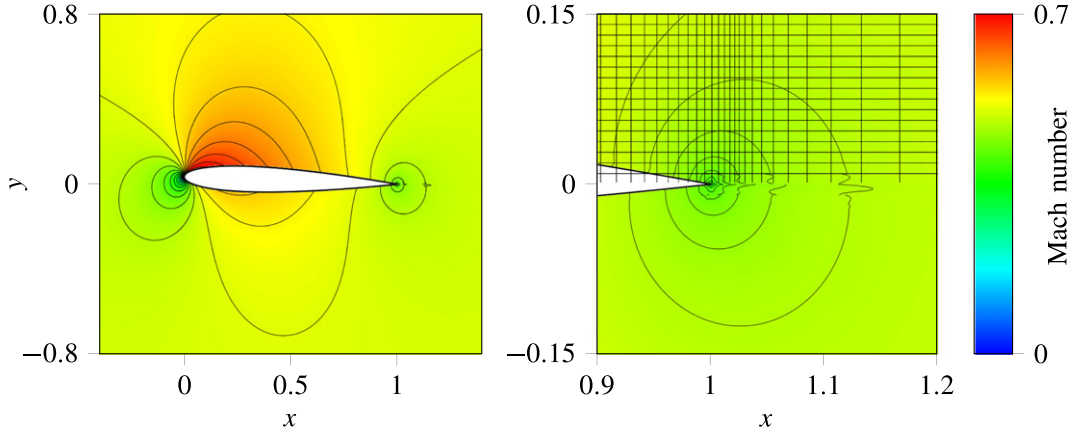


Figure 16. Zoom into the region around the NACA0012 airfoil for the medium mesh. Solid red line shows the zero contour of the level set function (49) for an airfoil with a rotation of $\alpha = 2°$.



(a) Mach number distribution and contours (b) Zoom into trailing edge

Figure 17. Inviscid subsonic flow around a NACA0012 airfoil on a fine mesh (128×128 cells) with $P = 3$.

DOI: 10.1002/nme

boundary. This practice avoids problems stemming from the smooth approximation of a non-smooth level set function, which is a general issue in the context of the level set method and thus also in the context of the numerical integration technique presented in Section 4.

In Figure 17, we show the Mach number contour (left) obtained on a fine mesh with $P = 3$. On the right, a zoom into the trailing edge with the corresponding grid resolution is shown. We observe smooth contours along the airfoil except for small wiggles at the trailing edge, which indicates that this region is still under-resolved.

The results are compared with values from literature in terms of the lift coefficient c_L and, in the following section, the drag coefficient c_D. These are defined as

$$c_L = \frac{2F_L}{\rho \vec{u}^2} \quad \text{and} \quad c_D = \frac{2F_D}{\rho \vec{u}^2}, \tag{50}$$

where F_L and F_D are the lift and drag forces acting on the surface of the airfoil. These forces can easily be calculated according to

$$\begin{pmatrix} F_D \\ F_L \end{pmatrix} = \oint \left(p\mathbf{I} + \mu \left[\nabla \vec{u} + \nabla \vec{u}^T - \frac{2}{3}(\nabla \cdot \vec{u})\mathbf{I} \right] \right) \vec{n}\, ds \tag{51}$$

by making use of the numerical integration technique presented in Section 4.

Computed lift coefficients c_L are summarized in Table III. Errors have been computed with respect to the reference solution given by Hartmann within [47]. Comparing the errors of the rotated to the non-rotated case, we see that both produce similar results. In addition, simulations on coarse meshes with higher polynomial degree produce smaller errors: The error in c_L is similar on the coarsest mesh for $P = 3$ (10 240 DOF) compared with the finest mesh with $P = 1$ (49 152 DOF), but with 80% less DOF. We conclude that it is beneficial to use higher order polynomial approximations in terms of accuracy per DOF, even though high-order convergence rates cannot be achieved because of the singularity at the trailing edge of the airfoil, which has previously been reported by Wang *et al.* [48].

In a next step, we study the computational cost of cut cells. We ran a sequence of test cases with different ratios of cut cells to uncut cells and averaged the runtime of a least 10 runs per configuration. The results indicate that cut cells are approximately 20 times more expensive than regular, uncut cells. This is not only due to the higher number of quadrature points, which leads to an increase of the cost by a factor of approximately 10. The remaining discrepancy is caused by the modified integration strategy with different quadrature node counts and distributions within the set of cut cells, which reduces caching efficiency and thus performance.

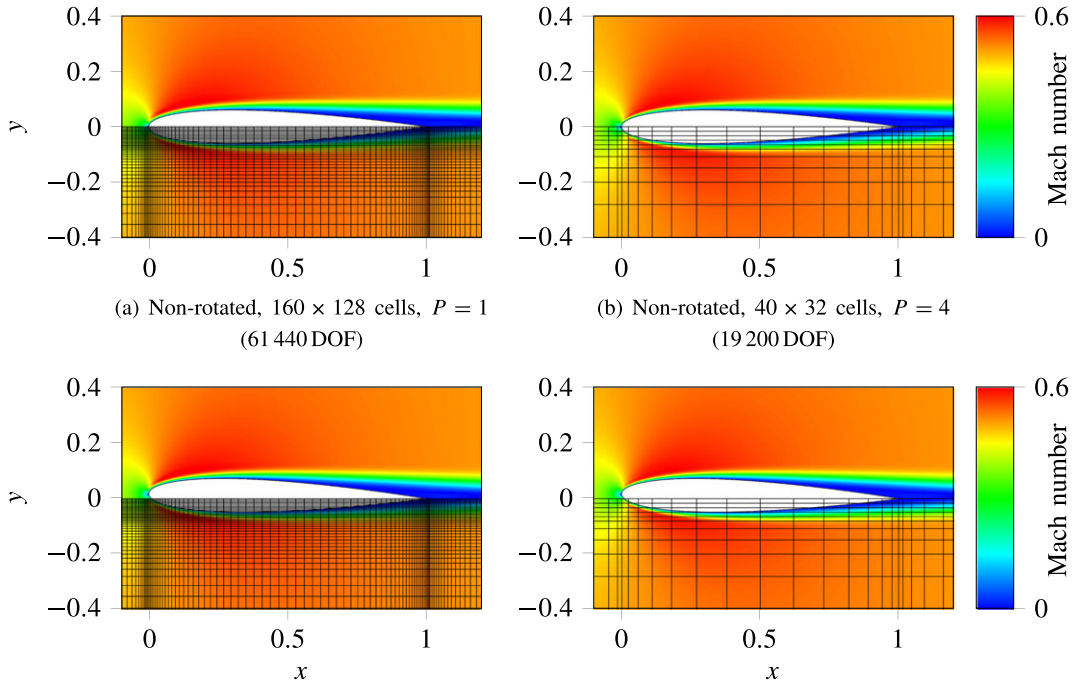
6.4. Viscous flow over a NACA0012 profile

Finally, we consider the viscous flow over a NACA0012 airfoil at Mach 0.5 with $\alpha = 1°$ and a Reynolds number of $\mathrm{Re} = 5000$ (based on the chord length). Equivalently to Wang *et al.* [48], we assume the viscosity to be constant. We apply far-field conditions everywhere except at the airfoil where we specify adiabatic wall boundary conditions with zero heat flux.

Table III. Inviscid subsonic flow past a NACA0012 airfoil: comparison of the error in the lift coefficient for the rotated and non-rotated case on different grids with varying polynomial orders.

Grid	Rotated	$P = 1$	$P = 2$	$P = 3$
32×32	×	$8.22 \cdot 10^{-02}$	$3.45 \cdot 10^{-02}$	$1.70 \cdot 10^{-02}$
	✓	$9.59 \cdot 10^{-02}$	$3.19 \cdot 10^{-02}$	$1.81 \cdot 10^{-02}$
64×64	×	$3.84 \cdot 10^{-02}$	$1.21 \cdot 10^{-02}$	$9.00 \cdot 10^{-03}$
	✓	$3.72 \cdot 10^{-02}$	$1.41 \cdot 10^{-02}$	$8.42 \cdot 10^{-03}$
128×128	×	$1.63 \cdot 10^{-02}$	$7.85 \cdot 10^{-03}$	$5.97 \cdot 10^{-03}$
	✓	$1.41 \cdot 10^{-02}$	$7.50 \cdot 10^{-03}$	$6.55 \cdot 10^{-03}$

DOI: 10.1002/nme



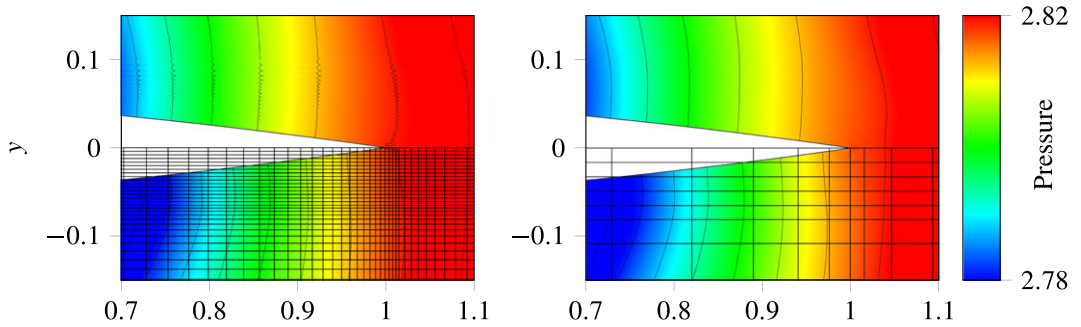
(a) Non-rotated, 160 × 128 cells, $P = 1$ (61 440 DOF)

(b) Non-rotated, 40 × 32 cells, $P = 4$ (19 200 DOF)

(c) Rotated, 160 × 128 cells, $P = 1$ (61 440 DOF)

(d) Rotated, 40 × 32 cells, $P = 4$ (19 200 DOF)

Figure 18. Comparison of the Mach number distribution in the non-rotated (top) and the rotated (bottom) cases for the viscous flow around a NACA0012 airfoil at different resolutions.



(a) 160 × 128 cells, $P = 1$ (61 440 DOF)

(b) 40 × 32 cells, $P = 4$ (19 200 DOF)

Figure 19. Comparison of the pressure distribution in the boundary layer around trailing edge for the non-rotated case.

Again, we analyze the differences between the rotated and non-rotated cases with respect to the lift coefficient c_L and the drag coefficient c_D (cf. Section 6.3). We study the solutions on a coarse mesh (1280 cells) with high order ($P = 4$) to a two times refined mesh (20,480 cells) with low order ($P = 1$). In Figure 18, a qualitative comparison reveals no visible difference between the two test cases, whether in the rotated or in the non-rotated case. A zoom into the trailing edge (cf. Figure 19) however reveals that the higher order simulation results in a smoother approximation of the boundary layer, despite its coarse resolution using only few cells. The corresponding results for the rotated case qualitatively are similar, which is why we omit them here.

Our numerical experiments indicate that the flow in the wake is slightly unsteady, which has also been reported by other groups using explicit time-stepping schemes (e.g., [49]). Moreover, the same phenomenon has been observed in [50] when solving the system with an inexact Newton-time method. More specifically, the author shows that large time-steps lead to a steady-state solution,

DOI: 10.1002/nme

Table IV. Viscous subsonic flow past a NACA0012 airfoil: error in the lift and drag coefficient computed on a coarse grid with high order and a two times finer grid with low order, additionally, a comparison of the rotated and non-rotated case.

	Rotated	40×32 cells $P = 4$	160×128 cells $P = 1$
Lift error	×	$5.26 \cdot 10^{-4}$	$6.89 \cdot 10^{-5}$
	✓	$6.26 \cdot 10^{-4}$	$8.48 \cdot 10^{-5}$
Drag error	×	$4.52 \cdot 10^{-5}$	$8.03 \cdot 10^{-3}$
	✓	$5.90 \cdot 10^{-5}$	$7.42 \cdot 10^{-5}$

while a higher temporal resolution leads to an unsteady solution. Swanson and Langer [51], on the other hand, provide evidence for a steady-state solution and claim that the level of *implicitness* in the algorithm is crucial to obtain a steady-state solution. Here, we do not discuss this issue any further because the effect on c_L and c_D is negligible.

A quantitative comparison is given in Table IV. In terms of c_D, the error for the high-order solution is almost two orders of magnitude lower. In the computation of c_L, on the other hand, the low-order simulation slightly outperforms the high-order simulation. We see again that the higher order solution leads to comparable results using 70% less DOF. Similar to the previous case, there is no significant disparity between the rotated and non-rotated cases.

7. CONCLUSION

We have presented a high-order discretization scheme for the compressible Euler and Navier–Stokes equations that is based on a cut cell DG method, where numerical integrals have been evaluated using a modified moment-fitting strategy. The scheme makes use of a non-intrusive cell-agglomeration strategy that strongly increases stable time-step sizes and alleviates problems with small and ill-shaped cut cells.

Robustness, accuracy, and convergence properties have been assessed in four two-dimensional test cases. The expected EOC of $P + 1$ could be obtained for both configurations featuring smooth exact solutions. Even though the agglomeration threshold δ does have negligible influence on this convergence rate, large values tend to increase absolute error levels. Despite the fact that the remaining test cases contain non-smooth regions near the sharp trailing edge, we observed a favorable accuracy per DOF for higher order simulations with $P \geqslant 2$.

Even though we have limited ourselves to two-dimensional test cases in the present work, the only challenge in an extension to the three-dimensional case lies in the generalization of the modified numerical integration procedure. In general, the moment-fitting approach is by no means limited to two space dimensions, which has, for example, been demonstrated in [38]. The bounding box strategy presented in Section 4.3 will however fail in three-dimensional settings where a face ε of a cell $\mathcal{K}$ is intersected by the interface, but *no* segment of $\partial\varepsilon$ contains a root of the level set function (i.e., where $\partial\varepsilon \cap \mathcal{I} = \emptyset$). Still, problematic intersections can be detected heuristically, for example, by checking the magnitude of φ and $\|\varphi\|$ in the vertices of ε [27, 29]. Troubled *faces* can then be subdivided until a non-problematic configuration for the definition of the bounding box is obtained. Rectifications of this type have, for example, been used in [20] and [32]. We are currently not aware of any other remedy for this drawback that does not involve an explicit representation of the problem geometry. While the subdivision of problematic faces is certainly a viable strategy, we are currently investigating more elegant solutions that avoid the added complexity.

A major advantage of the presented approach lies in its flexibility. In fact, the extension to viscous flows outlined in Section 6.4 only caused minimal changes in our implementation. While requiring a locally smooth level set function is disadvantageous for applications with non-smooth geometries, our strategy is by no means limited to applications with fixed immersed walls. Our future work will hence be focused on its extension to moving immersed boundaries with dynamic (de-)agglomeration of cut cells.

DOI: 10.1002/nme

APPENDIX A: CELL-AGGLOMERATION ALGEBRA

In this section, we derive relations for agglomerated cells following from definition (35), which have been used in Section 5.2. The Einstein summation convention is used for the sake brevity.

Equation (38) for the mass matrix of the agglomerated cell $\mathbf{M}^{\text{agg}}$ follows from

$$(\mathbf{M}^{\text{agg}})_{j,k} = \int_{\mathcal{K}^{\text{agg}}} \Phi_j^{\text{agg}} \Phi_k^{\text{agg}} dV \tag{52}$$

$$= \int_{\mathcal{K}^{\text{tar}}} \Phi_j^{\text{tar}} \Phi_k^{\text{tar}} dV + \int_{\mathcal{K}^{\text{src}}} (\vec{\Phi}^{\text{src}}\mathbf{Q})_j (\vec{\Phi}^{\text{src}}\mathbf{Q})_k dV \tag{53}$$

$$= (\mathbf{M}^{\text{tar}})_{j,k} + \int_{\mathcal{K}^{\text{src}}} \vec{\Phi}_l^{\text{src}}(\mathbf{Q})_{l,j} \vec{\Phi}_m^{\text{src}}(\mathbf{Q})_{m,k} dV \tag{54}$$

$$= (\mathbf{M}^{\text{tar}})_{j,k} + (\mathbf{Q})_{l,j} \left(\int_{\mathcal{K}^{\text{src}}} \Phi_l^{\text{src}} \Phi_m^{\text{src}} dV \right) (\mathbf{Q})_{m,k} \tag{55}$$

$$= (\mathbf{M}^{\text{tar}})_{j,k} + (\mathbf{Q}^T)_{j,l} (\mathbf{M}^{\text{src}})_{l,m} (\mathbf{Q})_{m,k}. \tag{56}$$

Equation (39) for the agglomerated solution follows from the orthogonality condition

$$\int_{\mathcal{K}^{\text{agg}}} (\tilde{c}^{\text{agg}} - \tilde{c}) \Phi_j^{\text{agg}} dV \stackrel{!}{=} 0 \tag{57}$$

for

$$\tilde{c} = \tilde{c}^{\text{tar}} + \tilde{c}^{\text{src}}, \tag{58}$$

leading to

$$(\mathbf{M}^{\text{agg}} \vec{\tilde{c}}^{\text{agg}})_j = \int_{\mathcal{K}^{\text{tar}}} \tilde{c}^{\text{tar}} \Phi_j^{\text{tar}} dV + \int_{\mathcal{K}^{\text{src}}} \tilde{c}^{\text{src}} (\vec{\Phi}^{\text{src}}\mathbf{Q})_j dV \tag{59}$$

$$= \tilde{c}_k^{\text{tar}} \int_{\mathcal{K}^{\text{tar}}} \Phi_k^{\text{tar}} \Phi_j^{\text{tar}} dV + \tilde{c}_k^{\text{src}} \int_{\mathcal{K}^{\text{src}}} \Phi_k^{\text{src}} (\vec{\Phi}^{\text{src}}\mathbf{Q})_j dV \tag{60}$$

$$= \tilde{c}_k^{\text{tar}} (\mathbf{M}^{\text{tar}})_{k,j} + \tilde{c}_k^{\text{src}} \int_{\mathcal{K}^{\text{src}}} \Phi_k^{\text{src}} \Phi_l^{\text{src}} (\mathbf{Q})_{l,j} dV \tag{61}$$

$$= \tilde{c}_k^{\text{tar}} (\mathbf{M}^{\text{tar}})_{k,j} + \tilde{c}_k^{\text{src}} (\mathbf{M}^{\text{src}})_{k,l} (\mathbf{Q})_{l,j} \tag{62}$$

$$= (\mathbf{M}^{\text{tar}})_{j,k} \tilde{c}_k^{\text{tar}} + (\mathbf{Q}^{\text{T}})_{j,l} (\mathbf{M}^{\text{src}})_{l,k} \tilde{c}_k^{\text{src}}, \tag{63}$$

where the last line exploits the symmetry of $\mathbf{M}^{\text{src}}$.

Equations (40) and (41) for the injection of the agglomerated solution into the non-agglomerated mesh follow from the orthogonality conditions

$$\int_{\mathcal{K}^{\text{tar}}} (\tilde{c}^{\text{tar}} - \tilde{c}^{\text{agg}}) \Phi_j^{\text{tar}} dV \stackrel{!}{=} 0 \tag{64}$$

and

DOI: 10.1002/nme

$$\int_{\mathcal{K}^{\mathrm{src}}} (\tilde{c}^{\mathrm{src}} - \tilde{c}^{\mathrm{agg}}) \Phi_j^{\mathrm{src}} dV \stackrel{!}{=} 0, \tag{65}$$

respectively. The first condition directly leads to

$$\tilde{c}^{\mathrm{tar}} = \tilde{c}^{\mathrm{agg}}, \tag{66}$$

while the second condition yields

$$\begin{aligned}
(\mathbf{M}^{\mathrm{src}} \vec{\tilde{c}}^{\mathrm{src}})_j &= \int_{\mathcal{K}^{\mathrm{src}}} \tilde{c}^{\mathrm{agg}} \Phi_j^{\mathrm{src}} dV && (67)\\
&= \int_{\mathcal{K}^{\mathrm{src}}} \Phi_k^{\mathrm{agg}} \tilde{c}^{\mathrm{agg}} \Phi_j^{\mathrm{src}} dV && (68)\\
&= \int_{\mathcal{K}^{\mathrm{src}}} (\vec{\Phi}^{\mathrm{src}} \mathbf{Q})_k \tilde{c}_k^{\mathrm{agg}} \Phi_j^{\mathrm{src}} dV && (69)\\
&= \int_{\mathcal{K}^{\mathrm{src}}} \Phi_l^{\mathrm{src}} (\mathbf{Q})_{l,k} \tilde{c}_k^{\mathrm{agg}} \Phi_j^{\mathrm{src}} dV && (70)\\
&= \left(\int_{\mathcal{K}^{\mathrm{src}}} \Phi_j^{\mathrm{src}} \Phi_l^{\mathrm{src}} dV \right) (\mathbf{Q})_{l,k} \tilde{c}_k^{\mathrm{agg}} && (71)\\
&= (\mathbf{M}^{\mathrm{src}})_{j,l} (\mathbf{Q})_{l,k} \tilde{c}_k^{\mathrm{agg}}. && (72)
\end{aligned}$$

Regarding Equation (44), we limit the discussion to the agglomeration of fluxes for uncut cells that are evaluated via Equation (15). The cut case (cf. Equation (25)) is completely analogous and does not give any new insights. Using the aforementioned results, we can verify that

$$\begin{aligned}
(\vec{g}^{\mathrm{agg}})_j &= \sum_{e=1}^{E} \int_{\varepsilon_e^{\mathrm{tar}}} g(\tilde{c}^{\mathrm{agg},-}, \tilde{c}^{\mathrm{agg},+}, \vec{n}^{\mathrm{tar}})\, \Phi_j^{\mathrm{agg}} dA - \int_{\mathcal{K}^{\mathrm{tar}}} \vec{f}(\tilde{c}^{\mathrm{agg}}) \cdot \nabla \Phi_j^{\mathrm{agg}} dV \\
&\quad + \sum_{e=1}^{E} \int_{\varepsilon_e^{\mathrm{src}}} g(\tilde{c}^{\mathrm{agg},-}, \tilde{c}^{\mathrm{agg},+}, \vec{n}^{\mathrm{src}})\, \Phi_j^{\mathrm{agg}} dA - \int_{\mathcal{K}^{\mathrm{src}}} \vec{f}(\tilde{c}^{\mathrm{agg}}) \cdot \nabla \Phi_j^{\mathrm{agg}} dV && (73)\\
&= \sum_{e=1}^{E} \int_{\varepsilon_e^{\mathrm{tar}}} g(\tilde{c}^{\mathrm{tar},-}, \tilde{c}^{\mathrm{tar},+}, \vec{n}^{\mathrm{tar}})\, \Phi_j^{\mathrm{tar}} dA - \int_{\mathcal{K}^{\mathrm{tar}}} \vec{f}(\tilde{c}^{\mathrm{tar}}) \cdot \nabla \Phi_j^{\mathrm{tar}} dV \\
&\quad + \sum_{e=1}^{E} \int_{\varepsilon_e^{\mathrm{src}}} g(\tilde{c}^{\mathrm{src},-}, \tilde{c}^{\mathrm{src},+}, \vec{n}^{\mathrm{src}})\, (\vec{\Phi}^{\mathrm{src}} \mathbf{Q})_j dA - \int_{\mathcal{K}^{\mathrm{src}}} \vec{f}(\tilde{c}^{\mathrm{src}}) \cdot \nabla (\vec{\Phi}^{\mathrm{src}} \mathbf{Q})_j dV && (74)\\
&= (\vec{g}^{\mathrm{tar}})_j \\
&\quad + \sum_{e=1}^{E} \int_{\varepsilon_e^{\mathrm{src}}} g(\tilde{c}^{\mathrm{src},-}, \tilde{c}^{\mathrm{src},+}, \vec{n}^{\mathrm{src}})\, \Phi_k^{\mathrm{src}} (\mathbf{Q})_{k,j} dA - \int_{\mathcal{K}^{\mathrm{src}}} \vec{f}(\tilde{c}^{\mathrm{src}}) \cdot \nabla \Phi_k^{\mathrm{src}} (\mathbf{Q})_{k,j} dV && (75)\\
&= (\vec{g}^{\mathrm{tar}})_j + (\vec{g}^{\mathrm{src}})_k (\mathbf{Q})_{k,j} && (76)
\end{aligned}$$

DOI: 10.1002/nme

holds. It is worth noting that the surface integral over the edge $\varepsilon^{\text{agg}} = \overline{\mathcal{K}^{\text{tar}}} \cap \overline{\mathcal{K}^{\text{src}}}$ between the agglomerated cells is considered twice in Equation (75). However, it does not contribute to $\vec{g}^{\text{agg}}$ because we require g to be consistent; that is,

$$g(\tilde{c}^{\text{tar},-}, \tilde{c}^{\text{tar},+}, \vec{n}^{\text{tar}}) = g(\tilde{c}^{\text{src},+}, \tilde{c}^{\text{src},-}, \vec{n}^{\text{tar}}) \tag{77}$$

$$= -g(\tilde{c}^{\text{src},-}, \tilde{c}^{\text{src},+}, \vec{n}^{\text{src}}) \tag{78}$$

on ε^{agg}.

ACKNOWLEDGEMENTS

The work of B. Müller, S. Krämer-Eis, and F. Kummer was supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at Technische Universität Darmstadt. The work of B. Müller was supported by the German Research Foundation (DFG) through Research Grant WA 2610/2-1. The work of F. Kummer was supported by the German DFG through Research Fellowship KU 2719/1-1. Some of the numerical results in this work have been obtained using the high performance computer Lichtenberg at the Technische Universität Darmstadt.

REFERENCES

1. Peskin CS. Flow patterns around heart valves : a digital computer method for solving the equations of motion. *Dissertation*, New York, 1972.
2. Mittal R, Iaccarino G. Immersed boundary methods. *Annual Review of Fluid Mechanics* 2005; **37**:239– 261.
3. Qin R, Krivodonova L. A discontinuous Galerkin method for solutions of the Euler equations on Cartesian grids with embedded geometries. *Journal of Computational Science* January 2013; **4**(1-2):24– 35.
4. Krivodonova L, Berger M. High-order accurate implementation of solid wall boundary conditions in curved geometries. *Journal of Computational Physics* January 2006; **211**(2):492– 512.
5. Fries T-P, Belytschko T. The extended/generalized finite element method: an overview of the method and its applications. *International Journal for Numerical Methods in Engineering* 2010; **84**(3):253– 304.
6. Parvizian J, Düster A, Rank E. Finite cell method: h- and p-extension for embedded domain problems in solid mechanics. *Computational Mechanics* 2007; **41**(1):121– 133.
7. Burman E, Claus S, Hansbo P, Larson MG, Massing A. CutFEM: discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering* December 2014; **104**(7):472– 501.
8. Brehm C, Hader C, Fasel HF. A locally stabilized immersed boundary method for the compressible Navier-Stokes equations. *Journal of Computational Physics* August 2015; **295**:475– 504.
9. Cheng KW, Fries T-P. Higher-order XFEM for curved strong and weak discontinuities. *International Journal for Numerical Methods in Engineering* 2010; **82**(5):564– 590.
10. Legrain G, Chevaugeon N, Dréau K. High order X-FEM and levelsets for complex microstructures: uncoupling geometry and approximation. *Computer Methods in Applied Mechanics and Engineering* October 2012; **241- 244**(0):172–189.
11. Massing A, Larson M, Logg A, Rognes arie E. A stabilized Nitsche fictitious domain method for the Stokes problem. *Journal of Scientific Computing* 2014; **61**(3):604– 628.
12. Brandstetter G, Govindjee S. A high-order immersed boundary discontinuous-Galerkin method for Poisson's equation with discontinuous coefficients and singular sources. *International Journal for Numerical Methods in Engineering* 2015; **101**(11):847– 869.
13. Fidkowski KJ, Darmofal DL. A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics* August 2007; **225**(2):1653– 1672.
14. Modisette JM, Darmofal DL. Toward a Robust, Higher-order Cut-cell Method for Viscous Flows, AIAA 2010 AIAA Meeting Papers, Orlando, Florida, 2010.
15. Sun H, Darmofal DL. An adaptive simplex cut-cell method for high-order discontinuous Galerkin discretizations of elliptic interface problems and conjugate heat transfer problems. *Journal of Computational Physics* 2014; **278**: 445– 468.
16. Bastian P, Engwer C. An unfitted finite element method using discontinuous Galerkin. *International Journal for Numerical Methods in Engineering* 2009; **79**(12):1557– 1576.
17. 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.
18. Johansson A, Larson MG. A high order discontinuous Galerkin Nitsche method for elliptic problems with fictitious boundary. *Numerische Mathematik* 2012; **123**(4):607– 628.

DOI: 10.1002/nme

19. Burman E, Hansbo P. Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method. *Applied Numerical Mathematics* 2012; **62**(4):328– 341.
20. Saye RI. High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles. *SIAM Journal on Scientific Computing* 2015; **37**(2):993– 1019.
21. Lehrenfeld C. High order unfitted finite element methods on level set domains using isoparametric mappings. *Computer Methods in Applied Mechanics and Engineering* 2016; **300**:716– 733.
22. Kummer Florian. Extended discontinuous Galerkin methods for two-phase flows: the spatial discretization. *International Journal for Numerical Methods in Engineering* 2016. DOI: 10.1002/nme.5288.
23. Hartmann R, Houston P. An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier-Stokes equations. *Journal of Computational Physics* 2008; **227**(22):9670– 9685.
24. Moës N, Dolbow J, Belytschko T. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering* 1999; **46**(1):131– 150.
25. Smereka P. The numerical approximation of a delta function with application to level set methods. *Journal of Computational Physics* 2006; **211**:77– 90.
26. Ventura G. On the elimination of quadrature subcells for discontinuous functions in the extended finite-element method. *International Journal for Numerical Methods in Engineering* April 2006; **66**(5):761– 795.
27. Min C, Gibou F. Geometric integration over irregular domains with application to level-set methods. *Journal of Computational Physics* October 2007; **226**(2):1432– 1443.
28. Zahedi S, Tornberg A-K. Delta function approximations in level set methods by distance function extension. *Journal of Computational Physics* 2010; **229**(6):2199– 2219.
29. Müller B, Kummer F, Oberlack M, Wang Y. Simple multidimensional integration of discontinuous functions with application to level set methods. *International Journal for Numerical Methods in Engineering* 2012; **92**(7):637– 651.
30. Sudhakar Y, Moitinho de Almeida JP, Wall WA. An accurate, robust, and easy-to-implement method for integration over arbitrary polyhedra: application to embedded interface methods. *Journal of Computational Physics* September 2014; **273**:393– 415.
31. Ventura G., Benvenuti E. Equivalent polynomials for quadrature in Heaviside function enriched elements. *International Journal for Numerical Methods in Engineering* May 2014; **102**(3-4):688– 710.
32. Fries T-P, Omerović S. Higher-order accurate integration of implicit geometries. *International Journal for Numerical Methods in Engineering* 2015.
33. Kudela L, Zander N, Bog T, Kollmannsberger S, Rank E. Efficient and accurate numerical quadrature for immersed boundary methods. *Advanced Modeling and Simulation in Engineering Sciences* 2015; **2**(1).
34. Wen X. High order numerical methods to two dimensional delta function integrals in level set methods. *Journal of Computational Physics* 2009; **228**(11):4273– 4290.
35. Bremer J, Gimbutas Z, Rokhlin V. A nonlinear optimization procedure for generalized Gaussian quadratures. *SIAM Journal on Scientific Computing* 2010; **32**(4):1761– 1788.
36. Mousavi SE, Sukumar N. Generalized Gaussian quadrature rules for discontinuities and crack singularities in the extended finite element method. *Computer Methods in Applied Mechanics and Engineering* December 2010; **199** (49–52):3237– 3249.
37. Mousavi SE, Sukumar N. Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons. *Computational Mechanics* 2011; **47**(5):535– 554.
38. 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.
39. Xiao H, Gimbutas Z. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Computers & Mathematics with Applications* January 2010; **59**(2):663– 676.
40. Bassi F, Rebay S. High-order accurate discontinuous finite element solution of the 2D Euler equations. *Journal of Computational Physics* 1997; **138**(2):251– 285.
41. Sudhakar Y, Wall WA. Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods. *Computer Methods in Applied Mechanics and Engineering* 2013; **258**:39– 54.
42. Hubrich S, Joulaian M, Düster A. Numerical integration in the finite cell method based on moment-fitting. *Proceedings of 3rd ECCOMAS Young Investigators Conference*, Aachen, Germany, 2015; 1– 4.
43. Toro EF. *Riemann Solvers and Numerical Methods for Fluid Dynamics* (3rd edn). Springer: Berlin/Heidelberg, 2009.
44. Massjung R. An unfitted discontinuous Galerkin method applied to elliptic interface problems. *SIAM Journal on Numerical Analysis* January 2012; **50**(6):3134– 3162.
45. Massing A, Larson MG, Logg A, Rognes ME. A stabilized Nitsche overlapping mesh method for the Stokes problem. *Numerische Mathematik* 2014; **128**(1):73–101.
46. Bassi F, Bartolo CD, Hartmann R, Nigro A. A discontinuous Galerkin method for inviscid low Mach number flows. *Journal of Computational Physics* 2009; **228**(11):3996– 4011.
47. Wang ZJ, Fidkowski K, Abgrall R, Bassi F, Caraeni D, Cary A, Deconinck H, Hartmann R, Hillewaert K, Huynh HT, Kroll N, May G, Persson P-O, Leer BV, Visbal M. *1st International Workshop on High-order CFD Methods*, 2013. http://zjwang.com/hiocfd.html [Accessed on 6 April 2016].

48. Wang Z, Fidkowski K, Abgrall R, Bassi F, Caraeni D, Cary A, Deconinck H, Hartmann R, Hillewaert K, Huynh H, Kroll N, May G, Persson P-O, van Leer B, Visbal M. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids* 2013:811– 845.
49. Gassner G, Lörcher F, Munz CD. A discontinuous Galerkin scheme based on a space-time expansion II. Viscous flow equations in multi dimensions. *Journal of Scientific Computing* 2008; **34**(3):260– 286.
50. Dolejší V. A design of residual error estimates for a high order BDF-DGFE method applied to compressible flows. *International Journal for Numerical Methods in Fluids* 2013; **73**(6):523– 559.
51. Swanson RC, Langer S. Steady-state laminar flow solutions for NACA 0012 airfoil. *Computers & Fluids* March 2016; **126**:102– 128.

DOI: 10.1002/nme