Journal of Computational Physics 302 (2015) 674–692

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A class of embedded discontinuous Galerkin methods for computational fluid dynamics

N.C. Nguyen [a,*,1], J. Peraire [a,1], B. Cockburn [b]

[a] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[b] School of Mathematics, University of Minnesota, Minneapolis, MN 55455, USA

ARTICLE INFO

Article history:
Received 24 August 2014
Received in revised form 9 June 2015
Accepted 12 September 2015
Available online 25 September 2015

Keywords:
Finite element methods
Discontinuous Galerkin methods
Hybrid/mixed methods
Euler equations
Navier–Stokes equations
Computational fluid dynamics

ABSTRACT

We present a class of embedded discontinuous Galerkin (EDG) methods for numerically solving the Euler equations and the Navier–Stokes equations. The essential ingredients are a local Galerkin projection of the underlying governing equations at the element level onto spaces of polynomials of degree k to parametrize the numerical solution in terms of the approximate trace, a judicious choice of the numerical flux to provide stability and consistency, and a global jump condition that weakly enforces the single-valuedness of the numerical flux to arrive at a global formulation in terms of the numerical trace. The EDG methods are thus obtained from the hybridizable discontinuous Galerkin (HDG) method by requiring the approximate trace to belong to smaller approximation spaces than the one in the HDG method. In the EDG methods, the numerical trace is taken to be continuous on a suitable collection of faces, thus resulting in an even smaller number of globally coupled degrees of freedom than in the HDG method. On the other hand, the EDG methods are no longer locally conservative. In the framework of convection–diffusion problems, this lack of local conservativity is reflected in the fact that the EDG methods do not provide the optimal convergence of the approximate gradient or the superconvergence for the scalar variable for diffusion-dominated problems as the HDG method does. However, since the HDG method does not display these properties in the convection-dominated regime, the EDG method becomes a reasonable alternative since it produces smaller algebraic systems than the HDG method. In fact, the resulting stiffness matrix has a similar sparsity pattern as that of the statically condensed continuous Galerkin (CG) method. The main advantage of the EDG methods is that they are generally more stable and robust than the CG method for solving convection-dominated problems. Numerical results are presented to illustrate the performance of the EDG methods. They confirm that, even though the EDG methods are not locally conservative, they are a viable alternative to the HDG method in the convection-dominated regime.

1. Introduction

Discontinuous Galerkin (DG) methods [3–5,19–23,28,29,31,34,35,38–41,57–59,61,64] have emerged as a competitive alternative for solving nonlinear hyperbolic systems of conservation laws because they possess some important advantages

* Corresponding author.
E-mail addresses: cuongng@mit.edu (N.C. Nguyen), peraire@mit.edu (J. Peraire), cockburn@math.umn.edu (B. Cockburn).
[1] This work was supported by AFOSR Grant No. FA9550-11-1-0141, AFOSR Grant No. FA9550-12-0357, and the Singapore-MIT Alliance for Research and Technology Centre.

http://dx.doi.org/10.1016/j.jcp.2015.09.024
0021-9991/© 2015 Elsevier Inc. All rights reserved.

over classical finite differences and finite volume methods. In particular, they can easily handle complicated geometries, have low dissipation, are locally conservative, high-order accurate, highly parallelizable, and more robust than continuous Galerkin (CG) methods for convection-dominated problems. However, in spite of these advantages, DG methods have not yet made a more significant impact for practical applications. This is largely due to the main criticism that DG methods are computationally expensive since they have too many global degrees of freedom. Indeed, the high computational cost and memory storage are a major impediment to the widespread application of DG methods for real-world problems. Therefore, it would be highly desirable to develop new DG methods that have all the advantages of DG methods and are computationally competitive with continuous Galerkin (CG) methods and finite volume methods.

In the spirit of making DG methods more competitive, researchers have developed more efficient DG methods such as the multiscale discontinuous Galerkin (MDG) method [37,7] and the embedded discontinuous Galerkin (EDG) method [32,14]. The MDG method was originally introduced in the framework of convection–diffusion problems [37], whereas the EDG in the framework of linear shells [32] and linear diffusion [14]. These two methods, which can give rise to identical schemes, are devised to solve for a globally continuous approximation of the solution (the numerical trace of the scalar variable for the MDG, and the numerical trace of the approximate displacement for the EDG) on the element boundaries. A brief comparison of the ideas upon which these methods are defined is given in [32].

The EDG methods are constructed by using a suitable modification of an associated method called a hybridizable discontinuous Galerkin (HDG) method. The modification is applied to the variational formulation defining the system of globally-coupled degrees of freedom and consists in reducing its size by simply using a strict subspace to define it. The HDG method was introduced in [14] in the framework of diffusion problems. It was analyzed in [11,16,18,24,25] where it is shown that it has many common features with the Raviart–Thomas (RT) mixed method [60] and the Brezzi–Douglas–Marini (BDM) mixed method [6]. In particular, in [16], it was proven that the HDG method using simplexes and polynomials of degree $k \geq 0$ in all the unknowns, provides approximations to the scalar variable and the flux which converge with the optimal order $k+1$ in the L^2-norm for any $k \geq 0$, and that the element-by-element averages of the scalar variable superconverges with order $k+2$ for $k \geq 1$; a local postprocessing scheme can then be used to obtain a new approximation to the scalar variable converging with order $k+2$ for $k \geq 1$. The EDG method constructed from this HDG method by using *continuous* approximate traces for the scalar variable was analyzed in [17]. Therein, it was shown that, although this results in a smaller system for the globally-coupled degrees of freedom, it also produces the loss of the local conservativity of the method. As a direct consequence, although the approximation for the scalar variable still converged with order $k+1$ for any $k \geq 0$, the approximation for the flux converges with the *suboptimal* order of k. Hence, the above-mentioned postprocessing converges only with order $k+1$ for $k \geq 1$. Numerical experiments indicated that the EDG method was in fact *less* efficient than its associated HDG method.

This disappointing result precluded further study of EDG methods associated to HDG methods with similar optimal convergence and superconvergence properties. This is the case for HDG methods for linear convection–diffusion problems [51, 12,8,9] and nonlinear convection–diffusion problems [12,52,65] in the diffusion-dominated regime, for HDG methods for the Stokes system of incompressible flow [13,15,47,53,26], for HDG methods for the incompressible Navier–Stokes equations [49, 50,54] in the diffusion-dominated regime, and for HDG methods for linear elasticity [26,27]. In particular, a unique feature of the HDG method for incompressible fluid flow is that the approximate velocity, pressure and velocity gradient converge with the optimal order $k+1$ in the L^2-norm for diffusion-dominated flows for any $k \geq 0$. Moreover, the element-by-element averages of the velocity superconverge and, a local postprocessing scheme proposed in [15,49] can be used to obtain a new approximate velocity which converges with order $k+2$ for $k \geq 1$.

On the other hand, it is reasonable to believe that, in the convection-dominated regime, all the above-mentioned HDG methods must behave like the classic DG methods in the pure convection limit. As a consequence, the optimality of the convergence, for example, to the gradient of the scalar variable, in the case of convection–diffusion problems is lost along with the above-mentioned superconvergence property. Numerical evidence of this fact is provided in [12]. It is then reasonable to expect that in this situation, the EDG methods might prove to be more efficient than the original HDG method. In other words, the EDG method [55] constructed from the HDG method for the compressible Euler and Navier–Stokes equations [44–46,56] in the convection-dominated regime, might be more efficient than the HDG method.

In this paper, we extend our previous work [55] to develop a class of EDG methods. The EDG methods are obtained from the HDG method by requiring the approximate trace to belong to smaller spaces than the one in the HDG method. In the EDG methods, the numerical trace is taken to be continuous on a suitable collection of faces, thus resulting in an even smaller number of globally coupled degrees of freedom than in the HDG method. Indeed, the original EDG method [55] produces a global matrix system that has the same sparsity pattern as that of the statically condensed continuous Galerkin (CG) method. As one instance of the class of EDG methods developed in this paper, the interior embedded DG (IEDG) method has slightly less globally coupled unknowns than the EDG method. Furthermore, the IEDG method enforces the boundary conditions more accurately than the EDG method. Numerical results presented herein confirm that the IEDG method outperforms the EDG method and thus establishes itself as a viable alternative to the HDG method.

The paper is organized as follows. In Section 2, we introduce the notation used throughout the paper and compare various DG methods in terms of the number of degrees of freedom and the number of nonzeros in their Jacobian matrix. We then introduce the class of EDG methods for the Euler equations in Section 3 and extend it to the compressible Navier–Stokes equations in Section 4. In Sections 3 and 4, we present numerical results to demonstrate the performance of the EDG methods. Finally, in Section 5, we end the paper with some concluding remarks.

2. Preliminaries

Throughout this paper we shall denote scalar variables by italic letters with no boldface (a, A, b, B, etc.), vector variables by italic boldface lowercase letters ($\boldsymbol{a}, \boldsymbol{b}$, etc.), and second-order tensor variables by italic boldface uppercase letters ($\boldsymbol{A}, \boldsymbol{B}$, etc.). The identity tensor shall be denoted by $\boldsymbol{I}$. The components of $\boldsymbol{a}$ and $\boldsymbol{A}$ shall be denoted as a_i and A_{ij}, respectively. The symbols, $\cdot$, $\times$, $\otimes$, shall denote the usual scalar product, vector product, and tensor product, respectively. We shall use boldface roman uppercase letters ($\mathbf{A}, \mathbf{B}$, etc.) to denote matrices with entries (A_{ij}, B_{ij}, etc.) and boldface roman lowercase letters ($\mathbf{a}, \mathbf{b}$, etc.) to denote column vectors with elements (a_i, b_i, etc.). We shall also denote sets and spaces by calligraphic letters ($\mathcal{A}, \mathcal{B}$, etc.). In this paper, the tensor product notation and matrix product notation are interchanged, that is, $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^T \boldsymbol{b}$, $\boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T$, $\boldsymbol{A} \cdot \boldsymbol{b} = \boldsymbol{A}\boldsymbol{b}$ and $\boldsymbol{A} \cdot \boldsymbol{B} = \boldsymbol{A}\boldsymbol{B}$.

2.1. Finite element mesh

Let Ω be a physical domain in $\mathbb{R}^d$ with Lipschitz boundary $\partial\Omega$ in $\mathbb{R}^{d-1}$. We denote by $\mathcal{T}_h$ a collection of disjoint elements (triangles and tetrahedra) that partition Ω. We also denote by $\partial\mathcal{T}_h$ the set $\{\partial K : K \in \mathcal{T}_h\}$, that is, the collection of the boundaries of all elements in $\mathcal{T}_h$. We shall denote by $\boldsymbol{n}$ the outward unit normal of ∂K. For an element K of the collection $\mathcal{T}_h$, $F = \partial K \cap \partial\Omega$ is the boundary face if the $(d-1)$-Lebesgue measure of F is nonzero. For two elements K^+ and K^- of the collection $\mathcal{T}_h$, $F = \partial K^+ \cap \partial K^-$ is the interior face between K^+ and K^- if the $(d-1)$-Lebesgue measure of F is nonzero. Let $\mathcal{E}_h^{\mathrm{I}}$ and $\mathcal{E}_h^{\partial}$ denote the set of interior and boundary faces, respectively. We denote by $\mathcal{E}_h$ the union of $\mathcal{E}_h^{\mathrm{I}}$ and $\mathcal{E}_h^{\partial}$. Note that by definition $\partial\mathcal{T}_h$ and $\mathcal{E}_h$ are different. More precisely, an interior face is counted twice in $\partial\mathcal{T}_h$ but once in $\mathcal{E}_h$ and a boundary face is counted once in both $\partial\mathcal{T}_h$ and $\mathcal{E}_h$.

2.2. Approximation spaces

Let $\mathcal{P}^k(D)$ denote the set of polynomials of degree at most k on a domain D. We introduce discontinuous finite element spaces

$$\begin{aligned}
\boldsymbol{\mathcal{U}}_h^k &= \{\boldsymbol{a} \in (L^2(\mathcal{T}_h))^m : \boldsymbol{a}|_K \in (\mathcal{P}^k(K))^m \, \forall K \in \mathcal{T}_h\}, \\
\boldsymbol{\mathcal{Q}}_h^k &= \{\boldsymbol{A} \in (L^2(\mathcal{T}_h))^{m\times d} : \boldsymbol{A}|_K \in (\mathcal{P}^k(K))^{m\times d} \, \forall K \in \mathcal{T}_h\},
\end{aligned} \tag{1}$$

for $\boldsymbol{a} = (a_i)$, $1 \le i \le m$, and $\boldsymbol{A} = (A_{ij})$, $1 \le i \le m$, $1 \le j \le d$. Here $L^2(D)$ is the space of square integrable functions on D. We further introduce traced finite element spaces

$$\begin{aligned}
\widehat{\boldsymbol{\mathcal{M}}}_h^k &= \{\boldsymbol{\mu} \in (L^2(\mathcal{E}_h))^m : \boldsymbol{\mu}|_F \in (\mathcal{P}^k(F))^m \, \forall F \in \mathcal{E}_h\}, \\
\widetilde{\boldsymbol{\mathcal{M}}}_h^k &= \{\boldsymbol{\mu} \in (C^0(\mathcal{E}_h))^m : \boldsymbol{\mu}|_F \in (\mathcal{P}^k(F))^m \, \forall F \in \mathcal{E}_h\},
\end{aligned} \tag{2}$$

for $\boldsymbol{\mu} = (\mu_i)$, $1 \le i \le m$, where $C^0(D)$ is the space of continuous functions on D. Note that $\widehat{\boldsymbol{\mathcal{M}}}_h^k$ consists of functions which are discontinuous over $\mathcal{E}_h$, whereas $\widetilde{\boldsymbol{\mathcal{M}}}_h^k$ consists of functions which are continuous over $\mathcal{E}_h$. Finally, we denote by $\boldsymbol{\mathcal{M}}_h^k$ a finite element approximation subspace that satisfies $\widetilde{\boldsymbol{\mathcal{M}}}_h^k \subset \boldsymbol{\mathcal{M}}_h^k \subset \widehat{\boldsymbol{\mathcal{M}}}_h^k$. In particular, we define

$$\boldsymbol{\mathcal{M}}_h^k = \{\boldsymbol{\mu} \in (L^2(\mathcal{E}_h))^m : \boldsymbol{\mu}|_F \in (\mathcal{P}^k(F))^m \, \forall F \in \mathcal{E}_h, \text{ and } \boldsymbol{\mu}|_{\mathcal{E}_h^{\mathrm{E}}} \in (C^0(\mathcal{E}_h^{\mathrm{E}}))^m\}, \tag{3}$$

where $\mathcal{E}_h^{\mathrm{E}}$ is a connected subset of $\mathcal{E}_h$.

Let us consider two important choices of $\boldsymbol{\mathcal{M}}_h^k$. The first choice is $\mathcal{E}_h^{\mathrm{E}} = \emptyset$ which implies $\boldsymbol{\mathcal{M}}_h^k = \widehat{\boldsymbol{\mathcal{M}}}_h^k$. This choice corresponds to the hybridizable discontinuous Galerkin (HDG) method [14]. The second choice is $\mathcal{E}_h^{\mathrm{E}} = \mathcal{E}_h$ which implies $\boldsymbol{\mathcal{M}}_h^k = \widetilde{\boldsymbol{\mathcal{M}}}_h^k$ and thus enforces the continuity of the approximate trace on *all faces*. This choice corresponds to the embedded discontinuous Galerkin (EDG) method introduced in [55]. Hence, the main difference between the HDG method and the EDG method lies in the definition of the approximation space for the approximate trace. This subtle difference is responsible for potentially important differences between the two methods in terms of efficiency and accuracy. As we discussed in the Introduction, the HDG method has been shown to be more accurate and efficient than the EDG method for diffusion problems, even though the EDG method has a smaller system of globally-coupled degrees of freedom than the HDG method, see [17], and we expect this advantage to hold in diffusion-dominated regimes, see [12]. However, in convection-dominated regimes this is no longer true, as we shall see by comparing the performance of the EDG to that of the HDG method proposed in [44,45,56].

Another interesting choice of the approximation space $\boldsymbol{\mathcal{M}}_h^k$ is obtained by setting $\mathcal{E}_h^{\mathrm{E}} = \mathcal{E}_h^{\mathrm{I}}$, which implies $\widetilde{\boldsymbol{\mathcal{M}}}_h^k \subset \boldsymbol{\mathcal{M}}_h^k \subset \widehat{\boldsymbol{\mathcal{M}}}_h^k$, where the inclusions are strict. The resulting approximation space consists of functions which are discontinuous over the union of the boundary faces $\mathcal{E}_h^{\partial}$ and continuous over the union of the interior faces $\mathcal{E}_h^{\mathrm{I}}$. The resulting method has a characteristic of the HDG method on the boundary faces and a characteristic of the EDG method on interior faces. Because the approximate trace is taken to be continuous only on the *interior faces*, we shall name this new method *interior* embedded

Table 1
Values of the coefficient α_{DOF} as a function of the number of spatial dimensions, the approximating polynomial order and the numerical discretization algorithm. This coefficient can be used in expression (4) to determine the total number of degrees of freedom in the problem.

	2D					3D				
Degree	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
DG	6	12	20	30	42	24	60	120	210	336
HDG	6	9	12	15	18	36	72	120	180	252
EDG	1	4	7	10	13	1	8	27	58	101
IEDG	<1	<4	<7	<10	<13	<1	<8	<27	<58	<101

DG (IEDG) method to distinguish it from the EDG method for which the trace is continuous on all faces, which, abusing the notation, we shall refer to simply as the EDG method. Thanks to the use of face-by-face local polynomial spaces on the domain boundary in the IEDG method, the degrees of freedom of the approximate trace on the boundary faces can be locally eliminated to yield a global matrix system involving the degrees of freedom of the numerical trace on the interior faces. As a result, the globally coupled unknowns of the IEDG method are even less than those of the EDG method. Furthermore, the IEDG method enforces boundary conditions more accurately than the EDG method. Hence, the IEDG method is more efficient and accurate than the EDG method.

We still need to introduce inner products associated with our finite element spaces. For functions a and b in $L^2(D)$, we denote $(a,b)_D = \int_D ab$ if D is a domain in $\mathbb{R}^d$ and $\langle a,b\rangle_D = \int_D ab$ if D is a domain in $\mathbb{R}^{d-1}$. Likewise, for functions $\boldsymbol{a}$ and $\boldsymbol{b}$ in $(L^2(D))^m$, we denote $(\boldsymbol{a},\boldsymbol{b})_D = \int_D \boldsymbol{a}\cdot\boldsymbol{b}$ if D is a domain in $\mathbb{R}^d$ and $\langle \boldsymbol{a},\boldsymbol{b}\rangle_D = \int_D \boldsymbol{a}\cdot\boldsymbol{b}$ if D is a domain in $\mathbb{R}^{d-1}$. For functions $\boldsymbol{A}$ and $\boldsymbol{B}$ in $(L^2(D))^{m\times d}$, we denote $(\boldsymbol{A},\boldsymbol{B})_D = \int_D \text{tr}(\boldsymbol{A}^T\boldsymbol{B})$ if D is a domain in $\mathbb{R}^d$ and $\langle \boldsymbol{A},\boldsymbol{B}\rangle_D = \int_D \text{tr}(\boldsymbol{A}^T\boldsymbol{B})$ if D is a domain in $\mathbb{R}^{d-1}$, where tr is the trace operator of a square matrix. We finally introduce the following volume inner products

$$(a,b)_{\mathcal{T}_h} = \sum_{K\in\mathcal{T}_h}(a,b)_K, \qquad (\boldsymbol{a},\boldsymbol{b})_{\mathcal{T}_h} = \sum_{K\in\mathcal{T}_h}(\boldsymbol{a},\boldsymbol{b})_K, \qquad (\boldsymbol{A},\boldsymbol{B})_{\mathcal{T}_h} = \sum_{K\in\mathcal{T}_h}(\boldsymbol{A},\boldsymbol{B})_K,$$

and boundary inner products

$$\langle a,b\rangle_{\partial\mathcal{T}_h} = \sum_{K\in\mathcal{T}_h}\langle a,b\rangle_{\partial K}, \qquad \langle \boldsymbol{a},\boldsymbol{b}\rangle_{\partial\mathcal{T}_h} = \sum_{K\in\mathcal{T}_h}\langle \boldsymbol{a},\boldsymbol{b}\rangle_{\partial K}, \qquad \langle \boldsymbol{A},\boldsymbol{B}\rangle_{\partial\mathcal{T}_h} = \sum_{K\in\mathcal{T}_h}\langle \boldsymbol{A},\boldsymbol{B}\rangle_{\partial K}.$$

These notations and definitions are necessary for the remainder of this paper.

2.3. Cost comparison with other DG methods

To put the EDG and IEDG methods in perspective, we compare them to the HDG method [44–46,56] and other DG methods such as the LDG method [28], the CDG method [57] or the method of Bassi and Rebay [4]. We consider triangular and tetrahedral discretizations and polynomial approximations of order $k = 1,\dots,5$ and calculate the number of globally coupled degrees of freedom as well as the number of non-zero elements in the Jacobian matrix. For implicit iterative solvers, the number of non-zero elements in the Jacobian matrix provides a good indication of the computational cost. We consider large meshes so that, if N_p is the number of vertices, the number of triangles in 2D is approximately $2N_p$ and the number of tetrahedra in 3D is $6N_p$. Even though for general 3D meshes, the ratio of tetrahedra to vertices can be unbounded, the above assumptions are reasonable for well shaped meshes and consistent with those presented in [36].

For a system of conservation laws involving N_c components (for the Euler and laminar the Navier–Stokes equations $N_c = 4$ in 2D and $N_c = 5$ in 3D), the total number of degrees of freedom is given by

$$DOF = N_p N_c \alpha_{\text{DOF}}, \tag{4}$$

where the coefficient α_{DOF} is given in Table 1. The total number of non-zero entries in the Jacobian matrix is given by

$$NNZ = N_p N_c^2 \alpha_{\text{NNZ}}. \tag{5}$$

The coefficient α_{NNZ} is given in Table 2. The detailed calculation of α_{DOF} and α_{NNZ} can be found in [36]. We note that the coefficients α_{DOF} and α_{NNZ} for the IEDG method cannot be determined precisely because they depend on the number of the interior faces and the number of the boundary faces. For very large meshes in which the number of interior faces is much larger than the number of boundary faces, the coefficients for the IEDG method will be close to those for the EDG method. For small meshes in which the number of interior faces is about the number of boundary faces, the coefficients for the IEDG method will be significantly less than those for the EDG method. The IEDG method always has smaller numbers of degrees of freedom and non-zero entries than the EDG method.

In all cases, we observe a dramatic reduction in computational cost when the EDG and IEDG methods are considered. We also note that the total number of degrees of freedom in HDG and EDG methods scales like k in 2D and k^2 in 3D. This compares very favorably to a scaling of k^2 and k^3 in 2D and 3D, respectively, for the DG method. If we look at the number

Table 2
Values of the coefficient α_{NNZ} as a function of the number of spatial dimensions, the approximating polynomial order and the numerical discretization algorithm. This coefficient can be used in expression (5) to determine the total number of degrees of freedom in the problem.

	2D					3D				
Degree	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
DG	72	288	800	1800	3528	480	3000	12 000	36 750	94 080
HDG	60	135	240	375	540	756	3024	8400	18 900	37 044
EDG	7	46	115	214	343	15	230	1311	4410	11 183
IEDG	<7	<46	<115	<214	<343	<15	<230	<1311	<4410	<11 183

of non-zero entries in the matrix, the scaling of the EDG and HDG methods is like k^2 in 2D and k^4 in 3D, whereas for the DG method, the scaling is like k^4 in 2D and k^6 in 3D.

3. The Euler equations

3.1. Governing equations

We consider the steady-state Euler equations of gas dynamics written in nondimensional conservation form as

$$\nabla \cdot \boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{f}, \quad \text{in } \Omega, \tag{6}$$

where $\boldsymbol{u} = (u_i)$, $1 \le i \le m = d+2$, is a vector of conserved dimensionless quantities (namely, density, momentum and energy), $\boldsymbol{F}(\boldsymbol{u})$ is the inviscid flux (a $m \times d$ matrix-valued function), and $\boldsymbol{f}$ is a source term. The Euler equations (6) must be supplemented with appropriate boundary conditions at the inflow and outflow boundaries and at the solid wall. We shall discuss these boundary conditions later.

3.2. The EDG methods

3.2.1. Weak formulation

We consider the governing equations (6) on any element $K \in \mathcal{T}_h$, multiply them with some test functions $\boldsymbol{w}$, and integrate the resulting equations by parts to obtain

$$-(\boldsymbol{F}(\boldsymbol{u}), \nabla \boldsymbol{w})_K + \langle \boldsymbol{F}(\widehat{\boldsymbol{u}}) \cdot \boldsymbol{n}, \boldsymbol{w} \rangle_{\partial K} = (\boldsymbol{f}, \boldsymbol{w})_K, \tag{7}$$

where $\widehat{\boldsymbol{u}}$ is the trace of the solution and $\boldsymbol{n}$ is the outward normal unit vector to ∂K.

Following [44,45,56] we introduce the so-called *local problem*: For any approximation $\widehat{\boldsymbol{u}}_h$ to the trace of the solution $\widehat{\boldsymbol{u}}$ on ∂K we find $\boldsymbol{u}_h \in \boldsymbol{\mathcal{U}}_h^k$ such that it satisfies

$$-(\boldsymbol{F}(\boldsymbol{u}_h), \nabla \boldsymbol{w})_K + \langle \boldsymbol{F}(\widehat{\boldsymbol{u}}_h) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{w} \rangle_{\partial K} = (\boldsymbol{f}, \boldsymbol{w})_K, \quad \forall \boldsymbol{w} \in (\mathcal{P}^k(K))^m. \tag{8}$$

Here $\boldsymbol{S}$ is the so-called stabilization matrix which is added to render the local problem (8) well-defined for any given value of $\widehat{\boldsymbol{u}}_h$ in a suitable finite element space. The local problem defines $\boldsymbol{u}_h$ as a function of $\widehat{\boldsymbol{u}}_h$.

We still need to determine $\widehat{\boldsymbol{u}}_h$. To this end, we require that $\widehat{\boldsymbol{u}}_h \in \boldsymbol{\mathcal{M}}_h^k$ satisfies

$$\langle \boldsymbol{F}(\widehat{\boldsymbol{u}}_h) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{\mu} \rangle_{\partial \mathcal{T}_h \backslash \partial\Omega} + \langle \boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h), \boldsymbol{\mu} \rangle_{\partial\Omega} = 0, \quad \forall \boldsymbol{\mu} \in \boldsymbol{\mathcal{M}}_h^k. \tag{9}$$

Here $\boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h)$ is the boundary numerical flux whose definition depends on the boundary conditions and will be given later.

Summing (8) over all elements and combining it with (9) we obtain that $(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h) \in \boldsymbol{\mathcal{U}}_h^k \times \boldsymbol{\mathcal{M}}_h^k$ satisfies

$$-(\boldsymbol{F}(\boldsymbol{u}_h), \nabla \boldsymbol{w})_{\mathcal{T}_h} + \langle \boldsymbol{F}(\widehat{\boldsymbol{u}}_h) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{w} \rangle_{\partial \mathcal{T}_h} = (\boldsymbol{f}, \boldsymbol{w})_{\mathcal{T}_h}, \quad \forall \boldsymbol{w} \in \boldsymbol{\mathcal{U}}_h^k, \tag{10a}$$

$$\langle \boldsymbol{F}(\widehat{\boldsymbol{u}}_h) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{\mu} \rangle_{\partial \mathcal{T}_h \backslash \partial\Omega} + \langle \boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h), \boldsymbol{\mu} \rangle_{\partial\Omega} = 0, \quad \forall \boldsymbol{\mu} \in \boldsymbol{\mathcal{M}}_h^k. \tag{10b}$$

This is the weak formulation that defines a class of EDG methods for the Euler equations. In particular, the EDG method is obtained by setting $\mathcal{E}_h^{\mathrm{E}} = \mathcal{E}_h$ in (3), whereas the IEDG method is obtained by setting $\mathcal{E}_h^{\mathrm{E}} = \mathcal{E}_h^{\mathrm{I}}$. It remains to define the boundary numerical flux and the stabilization matrix.

3.2.2. Boundary conditions

At the inlet section or outlet section of the flow, we define the boundary flux as

$$\boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h) = \boldsymbol{A}_n^+(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h) - \boldsymbol{A}_n^-(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_\infty - \widehat{\boldsymbol{u}}_h), \tag{11}$$

where $\boldsymbol{u}_\infty$ is the freestream value, $\boldsymbol{A}_n^{\pm} = (\boldsymbol{A}_n \pm |\boldsymbol{A}_n|)/2$ and $\boldsymbol{A}_n = [\partial \boldsymbol{F}(\boldsymbol{u})/\partial \boldsymbol{u}] \cdot \boldsymbol{n}$.

At the solid surface with slip condition, we impose zero normal velocity and extrapolate the density, the tangential velocity, and the energy. Hence, we set

$$\boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h) = \boldsymbol{g}(\boldsymbol{u}_h) - \widehat{\boldsymbol{u}}_h, \tag{12}$$

where $\boldsymbol{g}(\boldsymbol{u}_h)$ is as follows

$$g_1 = u_{h1}, \quad (g_2, \dots, g_{m-1}) = \boldsymbol{v}_h - (\boldsymbol{v}_h \cdot \boldsymbol{n})\boldsymbol{n}, \quad g_m = u_{hm}. \tag{13}$$

Here $\boldsymbol{v}_h = (u_{h2}/u_{h1}, \dots, u_{hm-1}/u_{h1})$ is the velocity component of $\boldsymbol{u}_h$. Note that since $(\boldsymbol{v}_h - (\boldsymbol{v}_h \cdot \boldsymbol{n})\boldsymbol{n}) \cdot \boldsymbol{n} = 0$ we have $\widehat{\boldsymbol{v}}_h \cdot \boldsymbol{n} = 0$ on the solid wall, where $\widehat{\boldsymbol{v}}_h = (\widehat{u}_{h2}/\widehat{u}_{h1}, \dots, \widehat{u}_{hm-1}/\widehat{u}_{h1})$ is the velocity component of $\widehat{\boldsymbol{u}}_h$.

3.2.3. Stabilization matrix

There are several possible choices for the stabilization matrix including the Roe scheme [63] and Lax–Friedrichs scheme [29]. For the Roe scheme, we have

$$\boldsymbol{S}(\widehat{\boldsymbol{u}}_h) = \boldsymbol{L}(\widehat{\boldsymbol{u}}_h)|\boldsymbol{\Lambda}(\widehat{\boldsymbol{u}}_h)|\boldsymbol{R}(\widehat{\boldsymbol{u}}_h), \tag{14}$$

where $\boldsymbol{L}$, $\boldsymbol{R}$, and $\boldsymbol{\Lambda}$ are the matrices of the left and right eigenvectors, and eigenvalues of the Jacobian matrix $[\partial \boldsymbol{F}(\widehat{\boldsymbol{u}}_h)/\partial \widehat{\boldsymbol{u}}_h] \cdot \boldsymbol{n}$, respectively. The second choice is the local Lax–Friedrichs scheme

$$\boldsymbol{S} = (|\widehat{\boldsymbol{v}}_h \cdot \boldsymbol{n}| + c(\widehat{\boldsymbol{u}}_h))\boldsymbol{I}, \tag{15}$$

where $c(\widehat{\boldsymbol{u}}_h)$ is the local sound speed and $\boldsymbol{I}$ is the identity matrix. The third choice is the global Lax–Friedrichs scheme

$$\boldsymbol{S} = \tau^g_{\max}\boldsymbol{I}, \tag{16}$$

where $\tau^g_{\max}$ is the global maximum speed of the system. The choice of the stabilization matrix becomes less important as k increases because the numerical dissipation, which is of the order of $O(h^{k+1})$, decreases rapidly as k increases. The local Lax–Friedrichs stabilization matrix (15) is used in all the numerical examples presented in this paper.

3.3. Implementation

Applying the Newton–Raphson method to linearize the nonlinear weak formulation (10), we obtain the following linear weak formulation in terms of the Newton increment $(\delta\boldsymbol{u}_h, \delta\widehat{\boldsymbol{u}}_h) \in \boldsymbol{\mathcal{U}}^k_h \times \boldsymbol{\mathcal{M}}^k_h$ as

$$a(\delta\boldsymbol{u}_h, \boldsymbol{w}) + b(\delta\widehat{\boldsymbol{u}}_h, \boldsymbol{w}) = f(\boldsymbol{w}), \quad \forall \boldsymbol{w} \in \boldsymbol{\mathcal{U}}^k_h, \tag{17a}$$

$$c(\delta\boldsymbol{u}_h, \boldsymbol{\mu}) + d(\delta\widehat{\boldsymbol{u}}_h, \boldsymbol{\mu}) = g(\boldsymbol{\mu}), \quad \forall \boldsymbol{\mu} \in \boldsymbol{\mathcal{M}}^k_h. \tag{17b}$$

Here the forms are given by

$$\begin{aligned}
a(\boldsymbol{v}, \boldsymbol{w}) &= -\big(\boldsymbol{F}'(\boldsymbol{u}_h)\boldsymbol{v}, \nabla\boldsymbol{w}\big)_{\mathcal{T}_h} + \langle \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)\boldsymbol{v}, \boldsymbol{w}\rangle_{\partial\mathcal{T}_h}, \\
b(\boldsymbol{\eta}, \boldsymbol{w}) &= \big\langle(\boldsymbol{F}'(\widehat{\boldsymbol{u}}_h)\cdot\boldsymbol{n} + \boldsymbol{S}'(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h) - \boldsymbol{S}(\widehat{\boldsymbol{u}}_h))\boldsymbol{\eta}, \boldsymbol{w}\big\rangle_{\partial\mathcal{T}_h}, \\
c(\boldsymbol{v}, \boldsymbol{\mu}) &= \langle \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)\boldsymbol{v}, \boldsymbol{\mu}\rangle_{\partial\mathcal{T}_h\backslash\partial\Omega} + \big\langle \boldsymbol{b}'(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h)\boldsymbol{v}, \boldsymbol{\mu}\big\rangle_{\partial\Omega}, \\
d(\boldsymbol{\eta}, \boldsymbol{\mu}) &= \big\langle(\boldsymbol{F}'(\widehat{\boldsymbol{u}}_h)\cdot\boldsymbol{n} + \boldsymbol{S}'(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h) - \boldsymbol{S}(\widehat{\boldsymbol{u}}_h))\boldsymbol{\eta}, \boldsymbol{\mu}\big\rangle_{\partial\mathcal{T}_h\backslash\partial\Omega} + \big\langle \boldsymbol{b}''(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h)\boldsymbol{\eta}, \boldsymbol{\mu}\big\rangle_{\partial\Omega}, \\
f(\boldsymbol{w}) &= (\boldsymbol{f}, \boldsymbol{w})_{\mathcal{T}_h} + (\boldsymbol{F}(\boldsymbol{u}_h), \nabla\boldsymbol{w})_{\mathcal{T}_h} - \langle \boldsymbol{F}(\widehat{\boldsymbol{u}}_h)\cdot\boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{w}\rangle_{\partial\mathcal{T}_h}, \\
g(\boldsymbol{\mu}) &= -\langle \boldsymbol{F}(\widehat{\boldsymbol{u}}_h)\cdot\boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{\mu}\rangle_{\partial\mathcal{T}_h\backslash\partial\Omega} - \langle \boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h), \boldsymbol{\mu}\rangle_{\partial\Omega},
\end{aligned} \tag{18}$$

for all $(\boldsymbol{v}, \boldsymbol{\eta})$ and $(\boldsymbol{w}, \boldsymbol{\mu})$ in $\boldsymbol{\mathcal{U}}^k_h \times \boldsymbol{\mathcal{M}}^k_h$. Note that $\boldsymbol{F}'$, $\boldsymbol{S}'$, $\boldsymbol{b}'$, and $\boldsymbol{b}''$ denote $\partial\boldsymbol{F}/\partial\boldsymbol{u}_h$, $\partial\boldsymbol{S}/\partial\widehat{\boldsymbol{u}}_h$, $\partial\boldsymbol{b}/\partial\boldsymbol{u}_h$, and $\partial\boldsymbol{b}/\partial\widehat{\boldsymbol{u}}_h$, respectively.

The linear weak formulation gives rise to the following linear system

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{pmatrix} \delta\mathbf{u} \\ \delta\widehat{\mathbf{u}} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}, \tag{19}$$

where $\delta\mathbf{u}$ and $\delta\widehat{\mathbf{u}}$ are the vectors of degrees of freedom of $\delta\boldsymbol{u}_h$ and $\delta\widehat{\boldsymbol{u}}_h$, respectively. It is important to note that the matrix $\mathbf{A}$ has a block-diagonal structure due to the discontinuous nature of the approximation spaces. Therefore, it can be inverted at the element level to yield a block-diagonal matrix $(\mathbf{A})^{-1}$. We can thus eliminate $\delta\mathbf{u}$ to obtain a reduced system in terms of $\delta\widehat{\mathbf{u}}$ as

$$\mathbf{K}\,\delta\widehat{\mathbf{u}} = \mathbf{r}, \tag{20}$$

where

$$\mathbf{K} = -\mathbf{C}(\mathbf{A})^{-1}\mathbf{B} + \mathbf{D}, \qquad \mathbf{r} = \mathbf{g} - \mathbf{C}(\mathbf{A})^{-1}\mathbf{f}. \tag{21}$$

This is the global linear system to be solved at every Newton iteration. It is important to point out that the size and connectivity of the Jacobian matrix $\boldsymbol{K}$ depends on the choice of the approximation space $\boldsymbol{\mathcal{M}}^k_h$. For both the HDG method and the IEDG method, because the degrees of freedom on each boundary face are only connected to the degrees of freedom on the other faces of the element containing that particular boundary face, we can further reduce the size of the linear system (20) by locally eliminating the degrees of freedom of $\delta\widehat{\boldsymbol{u}}_h$ on the boundary faces. However, this additional static condensation cannot be applied to the EDG method because the degrees of freedom on boundary faces are connected to those on several elements. We refer to Subsection 2.3 for estimates about the size and the number of non-zero entries in the Jacobian matrix for the HDG, EDG and IEDG methods.

In practice, to form the global Jacobian matrix **K**, we do not need to explicitly compute the matrices **A**, **B**, **C**, and **D**. Instead, we compute the elemental matrices and elemental vectors, and perform the standard finite element assembly to form the system (20). In particular, the elemental matrix and vector on an element $K \in \mathcal{T}_h$ are given by

$$\mathbf{K}^K = -\mathbf{C}^K(\mathbf{A}^K)^{-1}\mathbf{B}^K + \mathbf{D}^K, \qquad \mathbf{r}^K = \mathbf{g}^K - \mathbf{C}^K(\mathbf{A}^K)^{-1}\mathbf{f}^K, \tag{22}$$

where

$$\begin{aligned}
\mathbf{A}^K_{ij} &= -\left(\boldsymbol{F}'(\boldsymbol{u}_h)\boldsymbol{\phi}_j, \nabla\boldsymbol{\phi}_i\right)_K + \left\langle \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)\boldsymbol{\phi}_j, \boldsymbol{\phi}_i\right\rangle_{\partial K},\\
\mathbf{B}^K_{in} &= \left\langle (\boldsymbol{F}'(\widehat{\boldsymbol{u}}_h)\cdot\boldsymbol{n} + \boldsymbol{S}'(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h) - \boldsymbol{S}(\widehat{\boldsymbol{u}}_h))\boldsymbol{\zeta}_n, \boldsymbol{\phi}_i\right\rangle_{\partial K},\\
\mathbf{C}^K_{lj} &= \left\langle \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)\boldsymbol{\phi}_j, \boldsymbol{\zeta}_l\right\rangle_{\partial K\setminus\partial\Omega} + \left\langle \boldsymbol{b}'(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h)\boldsymbol{\phi}_j, \boldsymbol{\zeta}_l\right\rangle_{\partial K\cap\partial\Omega},\\
\mathbf{D}^K_{ln} &= \left\langle (\boldsymbol{F}'(\widehat{\boldsymbol{u}}_h)\cdot\boldsymbol{n} + \boldsymbol{S}'(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h) - \boldsymbol{S}(\widehat{\boldsymbol{u}}_h))\boldsymbol{\zeta}_n, \boldsymbol{\zeta}_l\right\rangle_{\partial K\setminus\partial\Omega} + \left\langle \boldsymbol{b}''(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h)\boldsymbol{\zeta}_n, \boldsymbol{\zeta}_l\right\rangle_{\partial K\cap\partial\Omega},\\
\mathbf{f}^K_i &= (\boldsymbol{f}, \boldsymbol{\phi}_i)_K + \left(\boldsymbol{F}(\boldsymbol{u}_h), \nabla\boldsymbol{\phi}_i\right)_K - \left\langle \boldsymbol{F}(\widehat{\boldsymbol{u}}_h)\cdot\boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{\phi}_i\right\rangle_{\partial K},\\
\mathbf{g}^K_l &= -\left\langle \boldsymbol{F}(\widehat{\boldsymbol{u}}_h)\cdot\boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{\zeta}_l\right\rangle_{\partial K\setminus\partial\Omega} - \left\langle \boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h), \boldsymbol{\zeta}_l\right\rangle_{\partial K\cap\partial\Omega},
\end{aligned} \tag{23}$$

for $i, j = 1, \dots, M$ and $l, n = 1, \dots, N$. Here $\boldsymbol{\phi}_j$, $j = 1, \dots, M$, are polynomial basis functions in the local space $(\mathcal{P}^k(K^*))^m$ and $\boldsymbol{\zeta}_n$, $n = 1, \dots, N$, are polynomial basis functions in the local space $(\mathcal{P}^k(F^*))^{m\times e}$, where e is the number of faces on one element. Note here that K^* is the master element and F^* is the master face.

The elemental matrices and vectors are then assembled into the global Jacobian matrix and vector as

$$\mathbf{K}_{I^K(l),I^K(n)} := \mathbf{K}_{I^K(l),I^K(n)} + \mathbf{K}^K_{ln}, \qquad \mathbf{r}_{I^K(n)} := \mathbf{r}_{I^K(n)} + \mathbf{r}^K_n, \qquad l, n = 1, \dots, N, \tag{24}$$

where, for each $K \in \mathcal{T}_h$, I^K is the element connectivity vector that contains the numberings of the degrees of freedom of $\widehat{\boldsymbol{u}}_h$ on ∂K. Note that the element connectivity vectors I^K for all $K \in \mathcal{T}_h$ depend on the choice of the approximation space $\mathcal{M}^k_h$. Therefore, as regards the implementation, the only difference among the HDG method, the EDG method and the IEDG method lies in the assembly of the elemental quantities into the global Jacobian matrix and vector.

Finally, we note that we use the same shape functions to represent both the numerical solution and the geometry. It means that the space of polynomials of degree k on the master element K^* is used to represent the shape of every element $K \in \mathcal{T}_h$. The element and face integrals in (23) are transformed into the integrals on the master element and the master face, respectively. The resulting integrals are then computed by using Gauss quadratures.

3.4. The unsteady case

We consider the extension of the EDG method to the unsteady Euler equations:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla\cdot\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{f}, \quad \text{in } \Omega\times(0, T], \tag{25}$$

with the initial condition $\boldsymbol{u}(t = 0) = \boldsymbol{u}_0$ and appropriate boundary conditions. We denote by $(\boldsymbol{u}^n_h, \widehat{\boldsymbol{u}}^n_h)$ the numerical approximations to $(\boldsymbol{u}(t^n), \widehat{\boldsymbol{u}}(t^n))$ at time $t^n = n\Delta t^n$, where Δt^n is a timestep size at level n.

Using the EDG method to discretize (25) in space and the backward Euler method to discretize the time derivative, we obtain that $(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h) \in \mathcal{U}^k_h \times \mathcal{M}^k_h$ satisfies

$$\left(\frac{\boldsymbol{u}^n_h}{\Delta t^n}, \boldsymbol{w}\right)_{\mathcal{T}_h} - \left(\boldsymbol{F}(\boldsymbol{u}^n_h), \nabla\boldsymbol{w}\right)_{\mathcal{T}_h} + \left\langle \boldsymbol{F}(\widehat{\boldsymbol{u}}^n_h)\cdot\boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}^n_h)(\boldsymbol{u}^n_h - \widehat{\boldsymbol{u}}^n_h), \boldsymbol{w}\right\rangle_{\partial\mathcal{T}_h} = \left(\boldsymbol{f}^n + \frac{\boldsymbol{u}^{n-1}_h}{\Delta t^n}, \boldsymbol{w}\right)_{\mathcal{T}_h}, \tag{26a}$$

$$\left\langle \boldsymbol{F}(\widehat{\boldsymbol{u}}^n_h)\cdot\boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}^n_h)(\boldsymbol{u}^n_h - \widehat{\boldsymbol{u}}^n_h), \boldsymbol{\mu}\right\rangle_{\partial\mathcal{T}_h\setminus\partial\Omega} + \left\langle \boldsymbol{b}(\boldsymbol{u}^n_h, \widehat{\boldsymbol{u}}^n_h), \boldsymbol{\mu}\right\rangle_{\partial\Omega} = 0, \tag{26b}$$

for all $(\boldsymbol{w}, \boldsymbol{\mu}) \in \mathcal{U}^k_h \times \mathcal{M}^k_h$. Since this system is similar to the system (10) for the steady-state case, we apply the same solution procedure described above to solve the problem (26) at every time step.

Higher-order time-stepping methods such as the backward difference formula schemes or the diagonally implicit Runge–Kutta methods [1] can also be used to discretize the time derivative in a similar fashion. We refer to [45,48] for a detailed discussion.

3.5. Numerical examples

3.5.1. Ringleb flow

We first consider the Ringleb flow to demonstrate the optimal accuracy of the EDG method. The Ringleb flow is an exact smooth solution of the Euler equations obtained using the hodograph method [10]. For any given (x, y), we first obtain the radial velocity V by solving the following nonlinear equation

$$(x - 0.5L^2) + y^2 = \frac{1}{4\rho^2 V^4},$$

where

$$c = \sqrt{1 - \frac{V^2}{5}}, \quad \rho = c^5, \quad L = \frac{1}{c} + \frac{1}{3c^3} + \frac{1}{5c^5} - \frac{1}{2}\ln\frac{1+c}{1-c}.$$

We then compute the exact solution as

$$\rho = c^5, \quad p = c^7/\gamma, \quad v_1 = V\cos(\theta), \quad v_2 = V\sin(\theta),$$

where

$$\psi = \sqrt{\frac{1}{2V^2} - (x - 0.5L)\rho}, \qquad \theta = \arcsin(\psi V).$$

Since the exact solution can be determined at any spatial point, we take the domain Ω to be $(-2, -1) \times (1, 2)$. The boundary condition is prescribed by setting the freestream value $\boldsymbol{u}_\infty$ to the exact solution on the boundary of the domain. We consider triangular meshes that are obtained by splitting a regular $n \times n$ Cartesian grid into $2n^2$ triangles. On these meshes, we use polynomials of degree k to represent all the approximate variables with a nodal basis.

We present the L^2 error and convergence rate of the numerical solution $\boldsymbol{u}_h$ as a function of h and k in Table 3 for the HDG method, Table 4 for the EDG method, and in Table 5 for the IEDG method. We observe that the approximate solution converges with the optimal order $k+1$ for all the methods. Although the EDG and IEDG methods have less globally coupled degrees of freedom than the HDG method, they yield results as accurate as the HDG method. Furthermore, we note that the coefficients α_{DOF} and α_{NNZ} quickly approach those values in Tables 1 and 2, respectively, as the mesh size decreases.

3.5.2. Inviscid flow past a channel with a smooth bump

This problem is a Benchmark case proposed in the recent Workshop on High-Order CFD Methods [66], which aims at testing high-order methods for the computation of internal flow with a high-order curved boundary representation. The computational domain is bounded between $x = -1.5$ and $x = 1.5$, and between the bump and $y = 0.8$. The bump is defined as $y = 0.0625e^{-25x^2}$. The inflow Mach number is 0.5 at zero angle of attack. In this subsonic flow problem, entropy is constant in the flow field. The L^2 norm of the entropy error is then used as the indicator of solution accuracy since the analytical solution is unknown.

We consider quadrilateral meshes obtained by refining the coarsest mesh (a 2×8 Cartesian grid) as shown in Fig. 1. On these meshes, we use polynomials of degree k to represent all the approximate variables with a nodal basis. The entropy error convergence is presented in Fig. 2. We see that all the three schemes converge optimally with order $k+1$. Both the EDG and IEDG methods yield errors and convergence rates equally well as the HDG method.

We report in Table 6 and Table 7 the number of degrees of freedom and the number of nonzero entries of the global linear system, and the CPU time to solve the system for each method. We clearly see that the IEDG method is more efficient than the HDG method and the EDG method because it requires less memory storage and computational time than the other two methods. However, the comparison is not fair for the HDG method because the MATLAB sparse direct solver used for solution of the linear system does not make use of the block structure of the HDG method. An implementation of a linear solver that takes advantage of the block structure will certainly improve the performance of the HDG method, see [33]. Furthermore, we note that each method takes the same CPU time to assemble the linear system. The CPU time to assemble the system is usually smaller than the CPU time to solve it [62].

3.5.3. Inviscid subsonic flow past a NACA0012 airfoil

This example involves inviscid subsonic flow over a NACA2012 airfoil at a free stream Mach number $M_\infty = 0.5$ and an angle of attack $\alpha = 2$ degrees. This problem is a Benchmark case proposed in the recent Workshop on High-Order CFD Methods [66], which aims at testing high-order methods for the computation of external flow with a high-order curved boundary representation. The far field boundary is placed at a distance of ten chords away from the airfoil. We refer to [66] for a detailed description of this test case.

We consider two different meshes shown in Fig. 3. The coarse mesh consists of $N_e = 256$ quadrilateral elements, while the fine mesh has $N_e = 1024$ quadrilateral elements. We use polynomials of degree $k = 4$ to represent both the numerical solution and geometry on the coarse mesh, while using polynomials of degree $k = 2$ to represent both the numerical solution and geometry on the fine mesh. In addition, we also compute a reference solution on a very fine mesh of 4096

Table 3
History of convergence of the HDG method for the Ringleb flow.

Mesh $h = 3/n$	$k = 1$				$k = 2$				$k = 3$				$k = 4$			
	error	order	α_{DOF}	α_{NNZ}	error	order	α_{DOF}	α_{NNZ}	error	order	α_{DOF}	α_{NNZ}	error	order	α_{DOF}	α_{NNZ}
3/2	4.35e−3	–	0.015	0.103	3.24e−4	–	0.022	0.231	2.35e−5	–	0.029	0.411	2.08e−6	–	0.037	0.643
3/4	1.10e−3	1.98	0.073	0.632	4.85e−5	2.74	0.110	1.421	1.43e−6	4.04	0.147	2.527	7.90e−8	4.72	0.184	3.949
1/8	2.80e−4	1.98	0.323	3.012	6.92e−6	2.81	0.485	6.777	8.63e−8	4.05	0.646	12.05	2.80e−9	4.82	0.808	18.82
3/16	7.06e−5	1.99	1.352	13.06	9.37e−7	2.88	2.028	29.39	5.18e−9	4.06	2.703	52.25	9.36e−11	4.90	3.379	81.63
3/32	1.77e−5	2.00	5.524	54.32	1.22e−7	2.94	8.287	122.2	3.83e−10	3.76	11.05	217.3	3.09e−12	4.92	13.81	339.5

Table 4
History of convergence of the EDG method for the Ringleb flow.

Mesh $h = 3/n$	$k = 1$				$k = 2$				$k = 3$				$k = 4$			
	error	order	α_{DOF}	α_{NNZ}	error	order	α_{DOF}	α_{NNZ}	error	order	α_{DOF}	α_{NNZ}	error	order	α_{DOF}	α_{NNZ}
3/2	4.31e−3	–	0.008	0.038	3.52e−4	–	0.023	0.199	2.47e−5	–	0.038	0.478	2.32e−6	–	0.052	0.875
3/4	1.08e−3	2.00	0.023	0.126	5.99e−5	2.56	0.074	0.736	1.50e−6	4.05	0.126	1.801	9.21e−8	4.65	0.177	3.321
3/8	2.76e−4	1.97	0.074	0.456	9.93e−6	2.59	0.265	2.822	8.74e−8	4.10	0.456	6.980	3.56e−9	4.69	0.647	12.93
3/16	7.00e−5	1.98	0.265	1.735	1.51e−6	2.72	1.000	11.05	5.14e−9	4.09	1.735	27.48	1.60e−10	4.48	2.469	51.01
3/32	1.75e−5	2.00	1.000	6.759	2.13e−7	2.83	3.880	43.73	3.62e−10	3.83	6.759	109.0	5.41e−12	4.89	9.369	202.6

Table 5
History of convergence of the IEDG method for the Ringleb flow.

Mesh $h = 3/n$	$k = 1$				$k = 2$				$k = 3$				$k = 4$			
	error	order	α_{DOF}	α_{NNZ}	error	order	α_{DOF}	α_{NNZ}	error	order	α_{DOF}	α_{NNZ}	error	order	α_{DOF}	α_{NNZ}
3/2	4.44e−3	–	0.006	0.028	3.65e−4	–	0.014	0.109	2.38e−5	–	0.021	0.241	2.23e−6	–	0.028	0.425
3/4	1.11e−3	2.00	0.021	0.117	6.13e−5	2.57	0.058	0.565	1.42e−6	4.06	0.095	1.329	1.04e−7	4.42	0.131	2.409
3/8	2.83e−4	1.97	0.073	0.447	1.02e−5	2.60	0.234	2.489	1.04e−7	3.78	0.396	6.037	3.97e−9	4.71	0.557	11.09
3/16	7.18e−5	1.98	0.264	1.725	1.54e−6	2.72	0.939	10.39	5.94e−9	4.13	1.615	25.59	1.45e−10	4.78	2.291	47.32
3/32	1.80e−5	2.00	0.998	6.750	2.37e−7	2.70	3.760	42.42	3.50e−10	4.08	6.522	105.3	5.10e−12	4.83	9.285	195.2

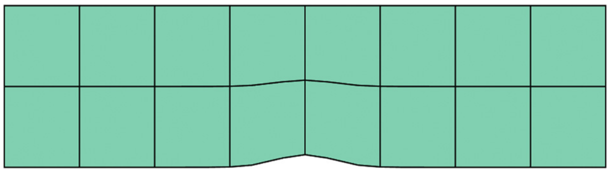

Fig. 1. Inviscid flow through a channel with a smooth bump: Initial mesh of 2 × 8 quadrilateral elements. Finer meshes are obtained by successively refined the initial mesh.

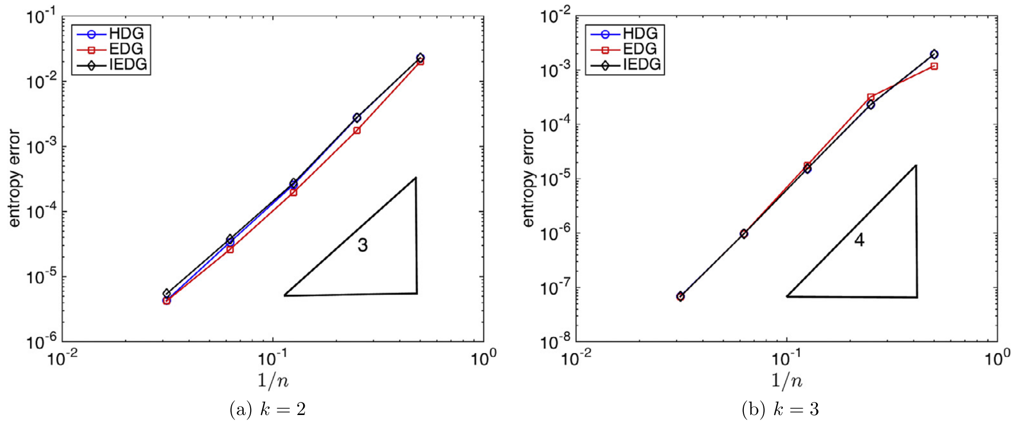

Fig. 2. Inviscid flow through a channel with a smooth bump: Entropy error norm versus mesh size, shown for (a) $k = 2$ and (b) $k = 3$.

Table 6
Inviscid flow through a channel with a smooth bump: Computational cost of solving the global linear system for the HDG, EDG, and IEDG methods for $k = 2$.

Mesh	Degrees of freedom			Number of non-zero entries			CPU time in seconds		
$n \times m$	HDG	EDG	IEDG	HDG	EDG	IEDG	HDG	EDG	IEDG
2 × 8	264	276	180	14 688	13 328	7952	0.0059	0.0035	0.0017
4 × 16	1296	932	756	92 736	50 704	40 528	0.0389	0.0124	0.0070
8 × 32	5664	3396	3060	442 368	197 648	177 872	0.1376	0.0580	0.0545
16 × 64	23 616	12 932	12 276	1 915 776	780 304	741 328	0.8954	0.2509	0.2359
64 × 128	96 384	50 436	49 140	7 959 168	3 100 688	3 023 312	3.9691	1.2546	1.2512

Table 7
Inviscid flow through a channel with a smooth bump: Computational cost of solving the global linear system for the HDG, EDG, and IEDG methods for $k = 3$.

Mesh	Degrees of freedom			Number of non-zero entries			CPU time in seconds		
$n \times m$	HDG	EDG	IEDG	HDG	EDG	IEDG	HDG	EDG	IEDG
2 × 8	352	444	268	26 112	31 344	16 816	0.0120	0.0114	0.0089
4 × 16	1728	1524	1188	164 864	120 528	91 920	0.0672	0.0341	0.0263
8 × 32	7552	5604	4948	786 432	472 464	415 696	0.1967	0.1143	0.1089
16 × 64	31 488	21 444	20 148	3 405 824	1 870 608	1 757 520	1.2723	0.6567	0.6460
64 × 128	128 512	83 844	81 268	14 149 632	7 443 984	7 218 256	10.5586	4.1901	4.1459

quadrilateral elements using polynomials of degree $k = 4$. We will use the reference solution to verify the accuracy of the numerical solutions computed on the coarse mesh and the fine mesh.

We show in Fig. 4 the Mach number contours for the numerical solutions computed using the IEDG method on the coarse mesh and the fine mesh. Note that the IEDG discretization on the coarse mesh with $k = 4$ has 6904 degrees of freedom and 771 040 nonzero entries in the Jacobian matrix, while the one on the fine mesh with $k = 2$ has 12 088 degrees of freedom and 729 568 nonzero entries. While the Mach number contours resemble, it is verified that the $k = 4$ solution is actually more accurate than the $k = 2$ solution since the former yields smaller errors in the lift and drag coefficients than the latter.

We report in Table 8 the errors in the lift coefficient and drag coefficient for the HDG, EDG, and IEDG methods. It can be seen that the discretizations on the coarse mesh with $k = 4$ produce smaller errors than those on the fine mesh with $k = 2$. This demonstrates the benefit of using polynomials of high degree to approximate the solution and geometry. Furthermore,

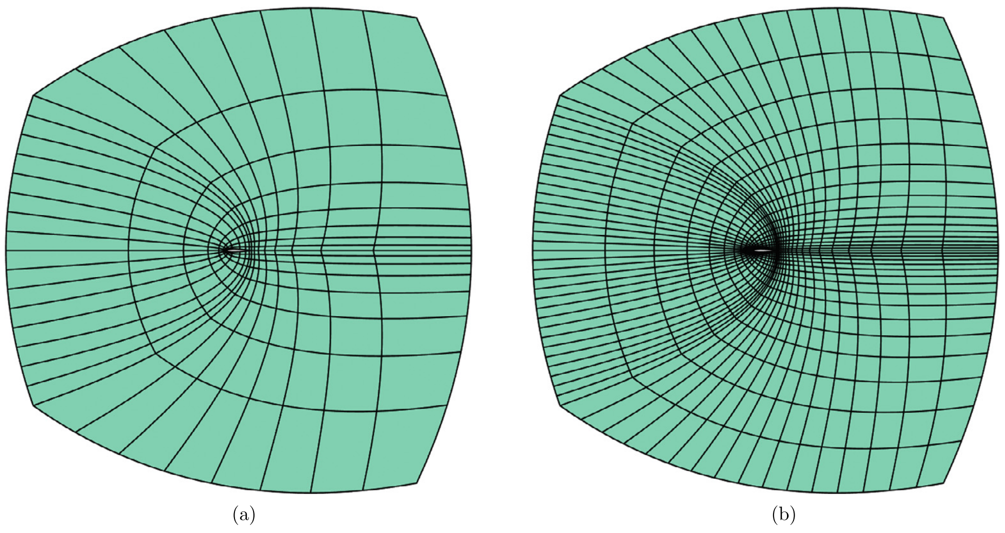

Fig. 3. Inviscid subsonic flow past a NACA0012 airfoil: (a) coarse mesh and (b) fine mesh.

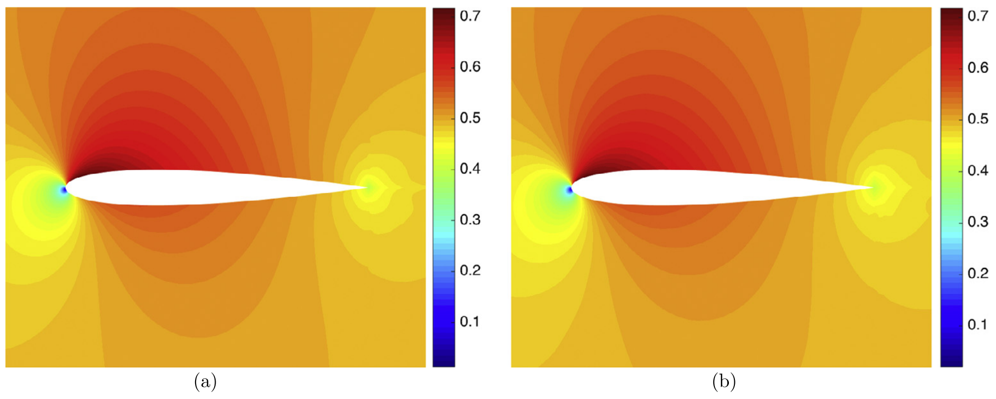

Fig. 4. Inviscid subsonic flow past a NACA0012 airfoil: Mach number contours for the numerical solutions computed using the IEDG method on (a) the coarse mesh with $k = 4$ and (b) the fine mesh with $k = 2$.

Table 8
Inviscid subsonic flow past a NACA0012 airfoil: Errors in the lift coefficient and drag coefficient for the HDG, EDG, and IEDG solutions computed on the coarse mesh with $k = 4$ and the fine mesh with $k = 2$.

Mesh (N_e, k)	Error in lift coefficient			Error in drag coefficient		
	HDG	EDG	IEDG	HDG	EDG	IEDG
$(256, 4)$	2.22×10^{-5}	2.95×10^{-4}	2.57×10^{-4}	8.93×10^{-5}	1.78×10^{-3}	2.11×10^{-4}
$(1024, 2)$	6.59×10^{-5}	3.80×10^{-4}	3.25×10^{-4}	1.53×10^{-4}	2.30×10^{-3}	3.07×10^{-4}

we observe that the HDG method is more accurate than the EDG and IEDG methods, while the IEDG method is more accurate than the EDG method.

Finally, we display in Fig. 5 the pressure coefficient and skin friction coefficient over the airfoil surface, which are obtained from the y-momentum and x-momentum components of the numerical flux $\boldsymbol{F}(\widehat{\boldsymbol{u}}_h) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h)$, respectively. The pressure coefficient distributions are very close to the reference one except that there are some small overshoots near the trailing edge due to a singularity at the trailing edge of the airfoil. Note that the exact value of the skin friction is zero for the inviscid flow. While both the HDG and IEDG methods have zero errors (up to machine tolerance) in the skin friction, the EDG method has quite large errors in the skin friction especially near the trailing edge as shown in Fig. 5. These results confirm that the IEDG method outperforms the EDG method.

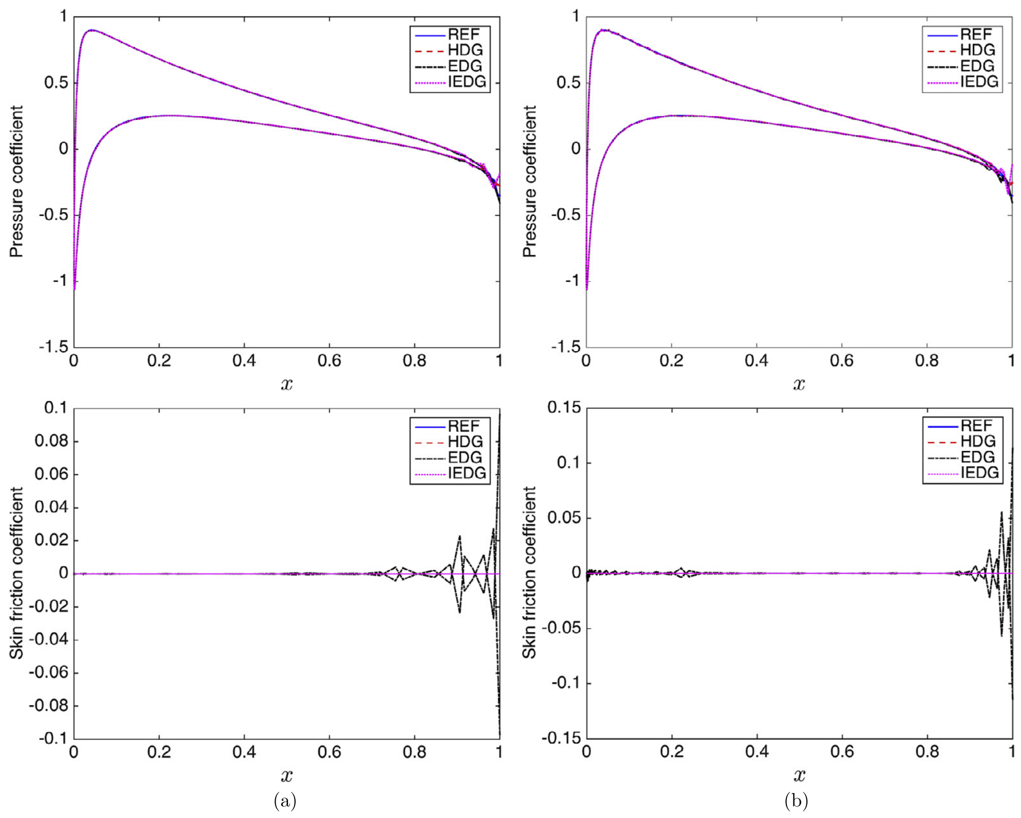

Fig. 5. Inviscid subsonic flow past a NACA0012 airfoil: Pressure coefficient (top) and skin friction coefficient (bottom) for (a) the coarse mesh with $k = 4$ and (b) the fine mesh with $k = 2$. Note that REF stands for the reference solution.

4. The Navier–Stokes equations

4.1. Governing equations

We consider the steady-state compressible Navier–Stokes equations in conservation form as

$$\nabla \cdot (\boldsymbol{F}(\boldsymbol{u}) + \boldsymbol{G}(\boldsymbol{u}, \nabla \boldsymbol{u})) = \boldsymbol{f}, \quad \text{in } \Omega, \tag{27}$$

where $\boldsymbol{u} = (u_i)$, $1 \le i \le m = d + 2$, is a vector of conserved dimensionless quantities (namely, density, momentum and energy), $\boldsymbol{F}(\boldsymbol{u})$ are inviscid fluxes of dimension $m \times d$, $\boldsymbol{G}(\boldsymbol{u}, \nabla \boldsymbol{u})$ are viscous fluxes of dimension $m \times d$, and $\boldsymbol{f}$ is a source term. The nondimensional form of the Navier–Stokes equations as well as the definition of the inviscid and viscous fluxes can be found in [2]. The Navier–Stokes equations (27) are supplemented with appropriate boundary conditions which will be discussed later.

4.2. The EDG methods

4.2.1. Formulation and implementation

We begin by considering the Navier–Stokes system (27) on any element $K \in \mathcal{T}_h$ and rewrite it as

$$\begin{aligned} \boldsymbol{q} - \nabla \boldsymbol{u} &= 0, \quad \text{in } K, \\ \nabla \cdot (\boldsymbol{F}(\boldsymbol{u}) + \boldsymbol{G}(\boldsymbol{u}, \boldsymbol{q})) &= \boldsymbol{f}, \quad \text{in } K. \end{aligned} \tag{28}$$

Multiplying with some test functions $(\boldsymbol{v}, \boldsymbol{w})$ and integrating the resulting equations by part we obtain

$$(\boldsymbol{q}, \boldsymbol{v})_K + (\boldsymbol{u}, \nabla \cdot \boldsymbol{v})_K - \langle \widehat{\boldsymbol{u}}, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} = 0, \tag{29a}$$

$$-(\boldsymbol{F}(\boldsymbol{u}) + \boldsymbol{G}(\boldsymbol{u}, \boldsymbol{q}), \nabla \boldsymbol{w})_K + \langle (\boldsymbol{F}(\widehat{\boldsymbol{u}}) + \boldsymbol{G}(\widehat{\boldsymbol{u}}, \boldsymbol{q})) \cdot \boldsymbol{n}, \boldsymbol{w} \rangle_{\partial K} = (\boldsymbol{f}, \boldsymbol{w})_K, \tag{29b}$$

where $\widehat{\boldsymbol{u}}$ is the trace of the solution and $\boldsymbol{n}$ is the outward normal unit vector to ∂K.

We next introduce the *local problem* on any element $K \in \mathcal{T}_h$ as: Find $(\boldsymbol{q}_h, \boldsymbol{u}_h) \in \mathcal{Q}_h^k \times \mathcal{U}_h^k$ such that it satisfies

$$\left(\boldsymbol{q}_h, \boldsymbol{v}\right)_K + (\boldsymbol{u}_h, \nabla \cdot \boldsymbol{v})_K - \langle \widehat{\boldsymbol{u}}_h, \boldsymbol{v} \cdot \boldsymbol{n}\rangle_{\partial K} = 0, \tag{30a}$$

$$-\left(\boldsymbol{F}(\boldsymbol{u}_h) + \boldsymbol{G}(\boldsymbol{u}_h, \boldsymbol{q}_h), \nabla \boldsymbol{w}\right)_K + \left\langle (\boldsymbol{F}(\widehat{\boldsymbol{u}}_h) + \boldsymbol{G}(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{w}\right\rangle_{\partial K} = (\boldsymbol{f}, \boldsymbol{w})_K, \tag{30b}$$

for all $(\boldsymbol{p}, \boldsymbol{w}) \in (\mathcal{P}^k(K))^{m\times d} \times (\mathcal{P}^k(K))^m$. This local problem defines $(\boldsymbol{q}_h, \boldsymbol{u}_h)$ locally as a function of $\widehat{\boldsymbol{u}}_h$. To determine $\widehat{\boldsymbol{u}}_h \in \mathcal{M}_h^k$ we require that

$$\left\langle (\boldsymbol{F}(\widehat{\boldsymbol{u}}_h) + \boldsymbol{G}(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{\mu}\right\rangle_{\partial \mathcal{T}_h \backslash \partial\Omega} + \left\langle \boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h), \boldsymbol{\mu}\right\rangle_{\partial\Omega} = 0, \quad \forall \boldsymbol{\mu} \in \mathcal{M}_h^k. \tag{31}$$

Here $\boldsymbol{S}(\widehat{\boldsymbol{u}}_h)$ is the stabilization matrix and $\boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)$ is the boundary numerical flux. The system (30)–(31) completely defines either the EDG method or the HDG method for the Navier–Stokes equations depending on the choice of the approximation space $\mathcal{M}_h^k$.

The implementation of the EDG method (or the HDG method) for the Navier–Stokes equations is similar to that of the EDG method (or the HDG method) for the Euler equations (6). We compute the elemental matrices and elemental vectors, and perform the standard finite element assembly to form the linear system at every Newton iteration. The elemental matrix and vector on an element $K \in \mathcal{T}_h$ are given by

$$\mathbf{K}^K = -\mathbf{C}^K (\mathbf{A}^K)^{-1} \mathbf{B}^K + \mathbf{D}^K, \qquad \mathbf{r}^K = \mathbf{g}^K - \mathbf{C}^K (\mathbf{A}^K)^{-1} \mathbf{f}^K, \tag{32}$$

where

$$\mathbf{A}^K = \begin{bmatrix} \mathbf{A}^{Kqq} & \mathbf{A}^{Kqu} \\ \mathbf{A}^{Kuq} & \mathbf{A}^{Kuu} \end{bmatrix}, \qquad \mathbf{B}^K = \begin{bmatrix} \mathbf{B}^{Kq} \\ \mathbf{B}^{Ku} \end{bmatrix}, \qquad \mathbf{C}^K = [\,\mathbf{C}^{Kq} \quad \mathbf{C}^{Ku}\,], \qquad \mathbf{f}^K = \begin{bmatrix} \mathbf{f}^{Kq} \\ \mathbf{f}^{Ku} \end{bmatrix}, \tag{33}$$

and

$$\begin{aligned}
\mathbf{A}^{Kqq}_{ps} &= \left(\boldsymbol{\psi}_s, \boldsymbol{\psi}_p\right)_K, \\
\mathbf{A}^{Kqu}_{pj} &= \left(\boldsymbol{\phi}_j, \nabla \cdot \boldsymbol{\psi}_p\right)_K, \\
\mathbf{A}^{Kuq}_{is} &= -\left(\boldsymbol{G}''(\boldsymbol{u}_h, \boldsymbol{q}_h)\boldsymbol{\psi}_s, \nabla \boldsymbol{\phi}_i\right)_K + \left\langle (\boldsymbol{G}''(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h) \cdot \boldsymbol{n})\boldsymbol{\psi}_s, \boldsymbol{\phi}_i\right\rangle_{\partial K}, \\
\mathbf{A}^{Kuu}_{ij} &= -\left((\boldsymbol{F}'(\boldsymbol{u}_h) + \boldsymbol{G}'(\boldsymbol{u}_h, \boldsymbol{q}_h))\boldsymbol{\phi}_j, \nabla \boldsymbol{\phi}_i\right)_K + \left\langle \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)\boldsymbol{\phi}_j, \boldsymbol{\phi}_i\right\rangle_{\partial K}, \\
\mathbf{B}^{Kq}_{pn} &= -\left\langle \boldsymbol{\zeta}_n, \boldsymbol{\psi}_p \cdot \boldsymbol{n}\right\rangle_{\partial K}, \\
\mathbf{B}^{Ku}_{in} &= \left\langle ((\boldsymbol{F}'(\widehat{\boldsymbol{u}}_h) + \boldsymbol{G}'(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)) \cdot \boldsymbol{n} + \boldsymbol{S}'(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h) - \boldsymbol{S}(\widehat{\boldsymbol{u}}_h))\boldsymbol{\zeta}_n, \boldsymbol{\phi}_i\right\rangle_{\partial K}, \\
\mathbf{C}^{Kq}_{ls} &= \left\langle (\boldsymbol{G}''(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h) \cdot \boldsymbol{n})\boldsymbol{\psi}_s, \boldsymbol{\zeta}_l\right\rangle_{\partial K\backslash\partial\Omega} + \left\langle \boldsymbol{b}'''(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)\boldsymbol{\psi}_s, \boldsymbol{\zeta}_l\right\rangle_{\partial K\cap\partial\Omega}, \\
\mathbf{C}^{Ku}_{lj} &= \left\langle \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)\boldsymbol{\phi}_j, \boldsymbol{\zeta}_l\right\rangle_{\partial K\backslash\partial\Omega} + \left\langle \boldsymbol{b}'(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)\boldsymbol{\phi}_j, \boldsymbol{\zeta}_l\right\rangle_{\partial K\cap\partial\Omega}, \\
\mathbf{D}^{K}_{ln} &= \left\langle ((\boldsymbol{F}'(\widehat{\boldsymbol{u}}_h) + \boldsymbol{G}'(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)) \cdot \boldsymbol{n} + \boldsymbol{S}'(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h) - \boldsymbol{S}(\widehat{\boldsymbol{u}}_h))\boldsymbol{\zeta}_n, \boldsymbol{\zeta}_l\right\rangle_{\partial K\backslash\partial\Omega} + \left\langle \boldsymbol{b}''(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)\boldsymbol{\zeta}_n, \boldsymbol{\zeta}_l\right\rangle_{\partial K\cap\partial\Omega}, \\
\mathbf{f}^{Kq}_{i} &= -\left(\boldsymbol{q}_h, \boldsymbol{\psi}_i\right)_K - \left(\boldsymbol{u}_h, \nabla \cdot \boldsymbol{\psi}_i\right)_K + \left\langle \widehat{\boldsymbol{u}}_h, \boldsymbol{\psi}_i \cdot \boldsymbol{n}\right\rangle_{\partial K}, \\
\mathbf{f}^{Ku}_{i} &= (\boldsymbol{f}, \boldsymbol{\phi}_i)_K + \left(\boldsymbol{F}(\boldsymbol{u}_h) + \boldsymbol{G}(\boldsymbol{u}_h, \boldsymbol{q}_h), \nabla \boldsymbol{\phi}_i\right)_K - \left\langle (\boldsymbol{F}(\widehat{\boldsymbol{u}}_h) + \boldsymbol{G}(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{\phi}_i\right\rangle_{\partial K}, \\
\mathbf{g}^{K}_{l} &= -\left\langle (\boldsymbol{F}(\widehat{\boldsymbol{u}}_h) + \boldsymbol{G}(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h), \boldsymbol{\zeta}_l\right\rangle_{\partial K\backslash\partial\Omega} - \left\langle \boldsymbol{b}(\boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h), \boldsymbol{\zeta}_l\right\rangle_{\partial K\cap\partial\Omega},
\end{aligned}$$

for $i, j = 1, \ldots, M$, $p, s = 1, \ldots, P$, and $l, n = 1, \ldots, N$. Note that $\boldsymbol{F}'$, $\boldsymbol{S}'$, $\boldsymbol{G}'$, $\boldsymbol{G}''$, $\boldsymbol{b}'$, $\boldsymbol{b}''$ and $\boldsymbol{b}'''$ denote $\partial \boldsymbol{F}/\partial \boldsymbol{u}_h$, $\partial \boldsymbol{S}/\partial \widehat{\boldsymbol{u}}_h$, $\partial \boldsymbol{G}/\partial \boldsymbol{u}_h$ (or $\partial \boldsymbol{G}/\partial \widehat{\boldsymbol{u}}_h$), $\partial \boldsymbol{G}/\partial \boldsymbol{q}_h$, $\partial \boldsymbol{b}/\partial \boldsymbol{u}_h$, $\partial \boldsymbol{b}/\partial \widehat{\boldsymbol{u}}_h$, and $\partial \boldsymbol{b}/\partial \widehat{\boldsymbol{q}}_h$, respectively. Here $\boldsymbol{\phi}_j$, $j = 1, \ldots, M$, are polynomial basis functions in the local space $(\mathcal{P}^k(K^*))^m$, $\boldsymbol{\psi}_p$, $p = 1, \ldots, P$, are polynomial basis functions in the local space $(\mathcal{P}^k(K^*))^{m\times d}$ and $\boldsymbol{\zeta}_n$, $n = 1, \ldots, N$, are polynomial basis functions in the local space $(\mathcal{P}^k(F^*))^{m\times e}$, where we recall that K^* is the master element and F^* is the master face.

Notice that the global Jacobian matrix for the Navier–Stokes equation also involves the degrees of freedom of $\widehat{\boldsymbol{u}}_h$ only and has the same structure as that for the Euler equations. Extension to the unsteady case is straightforward by using implicit time-stepping methods for temporal discretization as we already discussed in Subsection 3.4.

4.2.2. Wall boundary conditions

The inlet and outlet boundary conditions are imposed in the same manner as those of the Euler case. For the wall boundary conditions, we extrapolate the density, equate the fluid velocity to the wall velocity $\boldsymbol{v} = \boldsymbol{v}_w$, and set either a fixed temperature $T = T_w$ (isothermal wall) or a fixed heat flux $\partial T/\partial \boldsymbol{n} = q_w$ (adiabatic wall). The strategy to impose these conditions is described as follows. First, we set

$$b_1 = u_{h1} - \widehat{u}_{h1}, \tag{34}$$

which means that we extrapolate the density. We then set

$$b_i = v_{wi} - \widehat{u}_{hi}/\widehat{u}_{h1}, \quad 2 \le i \le m-1, \tag{35}$$

which means that we impose the no-slip condition at the solid wall. For the last component of $\boldsymbol{b}$, we need to distinguish between the isothermal wall and the adiabatic wall. For the isothermal wall, we set

$$b_m = T_w - T(\widehat{\boldsymbol{u}}_h), \tag{36}$$

where the approximate temperature $T(\widehat{\boldsymbol{u}}_h)$ is calculated based on $\widehat{\boldsymbol{u}}_h$. For the adiabatic wall, we set

$$b_m = q_w - q(\boldsymbol{q}_h, \boldsymbol{u}_h, \widehat{\boldsymbol{u}}_h), \tag{37}$$

where q is the m-th component of the numerical flux $(\boldsymbol{F}(\widehat{\boldsymbol{u}}_h) + \boldsymbol{G}(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h)) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h)$.

4.2.3. Stabilization matrix

To define the stabilization matrix for the Navier–Stokes equations, we add to the stabilization matrix in the EDG method and the HDG method for the Euler equations a viscous stabilization as

$$\boldsymbol{S}_v = \frac{\gamma}{Pr\,Re}\boldsymbol{I}. \tag{38}$$

Here *Re* is the Reynolds number, *Pr* is the Prandtl number, and γ is the heat capacity ratio. For compressible viscous flows the Reynolds number is typically very large and hence the contribution of the viscous component may be neglectable. However, the viscous stabilization may play an important role in cases where the Reynolds number is small.

4.3. Numerical examples

4.3.1. A linear convection–diffusion problem

The purpose of this example is to compare the proposed methods with the standard continuous Galerkin (CG) method. For this purpose, we consider the following linear convection–diffusion problem

$$\begin{aligned} \nabla \cdot (\boldsymbol{c}u) - \nabla^2 u = f, &\quad \text{in } \Omega, \\ u = 0 &\quad \text{on } \partial\Omega. \end{aligned} \tag{39}$$

Here $\Omega = (0,1) \times (0,1)$, $\mathbf{c} = (c_x, c_y)$, and the source term f is chosen to yield the following exact solution

$$u(x,y) = xy\frac{(1-e^{(x-1)c_x})(1-e^{(y-1)c_y})}{(1-e^{-c_x})(1-e^{-c_y})}.$$

The solution develops boundary layers along the boundaries $x = 1$ and $y = 1$ for large values of the velocity $\boldsymbol{c}$. This example serves to validate the performance of the proposed methods in the weakly convection-dominated regime. For this purpose we consider $\boldsymbol{c} = (20, 20)$ in our numerical experiments.

We consider triangular meshes obtained by splitting a regular $n \times n$ Cartesian grid into a total of $2n^2$ triangles, giving uniform element sizes of $h = 1/n$. On these meshes, we consider solutions of polynomial degree k represented using a nodal basis within each element, with the nodes uniformly distributed.

We present the error and order of convergence in L^2-norm in Table 9 for the HDG, EDG, EDG$_0$, IEDG, and CG methods. Here EDG$_0$ represents the EDG method which strongly imposes the Dirichlet boundary condition by setting the approximate trace to zero on the domain boundary. For the HDG method the L^2 error is computed using the postprocessed solution which is obtained by locally postprocessing the HDG solution [52]. As expected, the postprocessed solution of the HDG method converges with order $k + 2$, which is one order higher than the other methods. It is interesting to see that how the Dirichlet condition is enforced can affect the convergence rate of the EDG method especially for $k = 1$. In particular, for $k = 1$, strongly enforcing the Dirichlet condition yields optimal convergence rate of $k + 1$, whereas weakly imposing it results in suboptimal convergence of order k. However, both approaches yield optimal convergence rate of $k + 1$ for $k > 1$.

Furthermore, we see that both the IEDG method and the CG method deliver optimal convergence of order $k + 1$ for the numerical solution. However, the IEDG method yields smaller errors than the CG method, while both of them have the same number of global degrees of freedom and the same number of nonzero entries in the matrix system. It is known that the CG method for convection–diffusion problems produces overshoots when it does not have enough resolution to resolve the boundary layer feature. To demonstrate this point, we show in Fig. 6 the numerical solution computed using the IEDG method and the CG method computed for $n = 8$ and $k = 2$. We observe that the numerical solution of the CG method has considerable oscillations near the top and right boundaries, whereas the numerical solution of the IEDG method is quite smooth and clean. Indeed, as shown in Table 9, the IEDG solution has about 4 times smaller error than the CG solution for $n = 8$ and $k = 2$. The results presented here illustrate that the IEDG method is more robust and accurate than the CG method.

Table 9
Comparison of the methods in terms of the L^2 error in the solution and order of convergence. The postprocessed solution is used to compute the error for the HDG method. Note that EDG$_0$ represents the EDG method which strongly imposes the boundary condition by setting the approximate trace to zero on the domain boundary.

Degree	Mesh	HDG		EDG		EDG$_0$		IEDG		CG	
k	$h=1/n$	error	order	error	order	error	order	error	order	error	order
1	1/8	8.18e−3	–	2.57e−2	–	2.79e−2	–	1.42e−2	–	4.45e−2	–
	1/16	1.74e−3	2.23	2.27e−2	0.18	9.48e−3	1.56	5.03e−3	1.50	1.26e−2	1.83
	1/32	2.80e−4	2.63	1.72e−2	0.40	2.63e−3	1.85	1.75e−3	1.53	3.25e−3	1.95
	1/64	3.92e−5	2.84	1.10e−2	0.65	6.64e−4	1.99	5.36e−4	1.70	8.19e−4	1.99
	1/128	5.16e−6	2.93	6.27e−3	0.81	1.66e−4	2.00	1.49e−4	1.84	2.05e−4	2.00
2	1/8	1.52e−3	–	3.43e−3	–	4.28e−3	–	2.12e−3	–	7.97e−3	–
	1/16	1.60e−4	3.25	8.08e−4	2.08	7.12e−4	2.59	4.63e−4	2.19	1.29e−3	2.62
	1/32	1.21e−5	3.73	1.21e−4	2.74	9.4e−5	2.92	7.63e−5	2.60	1.73e−4	2.90
	1/64	8.02e−7	3.91	1.58e−5	2.93	1.18e−5	2.99	1.07e−5	2.83	2.20e−5	2.98
	1/128	5.12e−8	3.97	2.09e−6	2.92	1.49e−6	2.99	1.42e−6	2.92	2.76e−6	3.00
3	1/8	2.82e−4	–	7.89e−4	–	9.64e−4	–	5.28e−4	–	1.52e−3	–
	1/16	1.53e−5	4.20	1.06e−4	2.90	8.43e−5	3.52	5.62e−5	3.23	1.34e−4	3.50
	1/32	5.81e−7	4.72	9.78e−6	3.44	5.51e−6	3.94	4.42e−6	3.67	9.02e−6	3.90
	1/64	1.92e−8	4.92	7.25e−7	3.75	3.31e−7	4.06	2.95e−7	3.90	5.56e−7	4.02
	1/128	6.08e−10	4.98	4.87e−8	3.90	1.98e−8	4.06	1.87e−8	3.98	3.39e−8	4.04

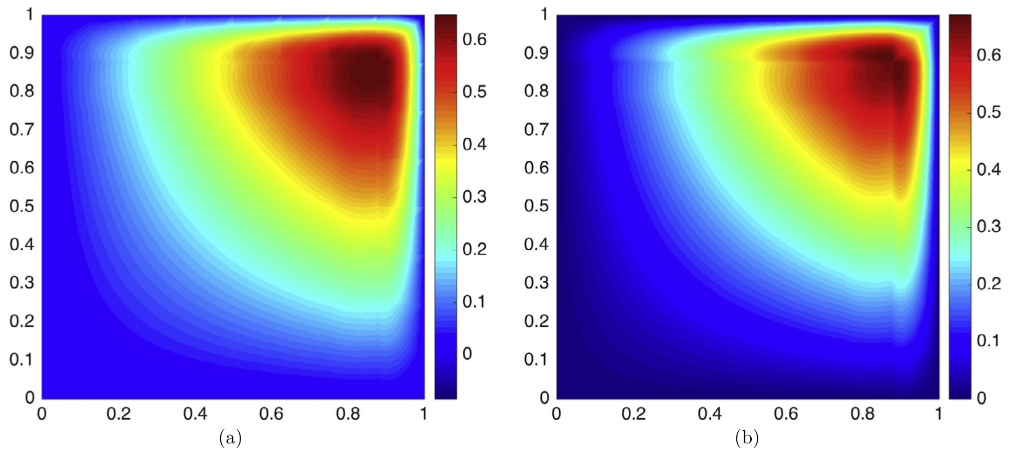

Fig. 6. Plots of the approximate solution computed using (a) the IEDG method and (b) the CG method for the mesh $n=8$ and the polynomial degree $k=2$.

4.3.2. Viscous subsonic flow past a NACA0012 airfoil

We consider the viscous analog of the inviscid case described in Subsection 3.5.3. The problem is now a viscous subsonic flow over a NACA2012 airfoil at a free stream Mach number $M_\infty = 0.5$, Reynolds number $Re = 5000$, and an angle of attack $\alpha = 2$ degree. We construct a sequence of C meshes. Each C mesh has $m \times n$ quadrilateral elements, where m denotes the number of divisions along the radial direction and n denotes the number of divisions along the airfoil surfaces and wakes. Fig. 7 shows the coarsest C mesh of $N_e = 4 \times 16$ ($m = 4$, $n = 16$) quadrilateral elements and the first refinement ($m = 8$, $n = 32$) of the coarsest mesh. The second refinement ($m = 16$, $n = 64$) is obtained by refining the first refinement. In addition, the third refinement ($m = 32$, $n = 128$) is only used to obtain the reference solution using the HDG method with polynomials of degree $k = 4$.

We show in Fig. 8 the Mach number of the numerical solutions computed using the IEDG method on the coarsest mesh with $k = 4$ and the first refinement with $k = 2$. Note that the IEDG discretization on the coarse mesh with $k = 4$ has 1656 degrees of freedom and 173 536 nonzero entries in the Jacobian matrix, while the one on the fine mesh with $k = 2$ has 3064 degrees of freedom and 180 960 nonzero entries. While the Mach numbers look similar, it can be seen from Fig. 9 that the $k = 4$ solution is actually more accurate than the $k = 2$ solution since the former yields smaller errors in the pressure coefficient and skin friction coefficient than the latter. Fig. 9 depicts the pressure coefficient and skin friction coefficient over the airfoil surface, which are obtained from the y-momentum and x-momentum components of the numerical flux $\boldsymbol{F}(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h) \cdot \boldsymbol{n} + \boldsymbol{S}(\widehat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h)$, respectively. We see that the $k = 4$ solutions computed on the coarsest mesh yield the pressure coefficient and skin friction coefficient distributions very close to the reference ones, while the $k = 2$ solutions computed on the first refinement produce some small oscillations in the pressure coefficient distribution near the leading edge. These results clearly demonstrate the benefit of using high-order discretizations.

Finally, we report in Table 10 the errors in the lift coefficient and drag coefficient for the HDG, EDG, and IEDG methods for polynomial degrees from 1 to 4. We observe that the errors decrease rapidly as the polynomial degree k increases, which

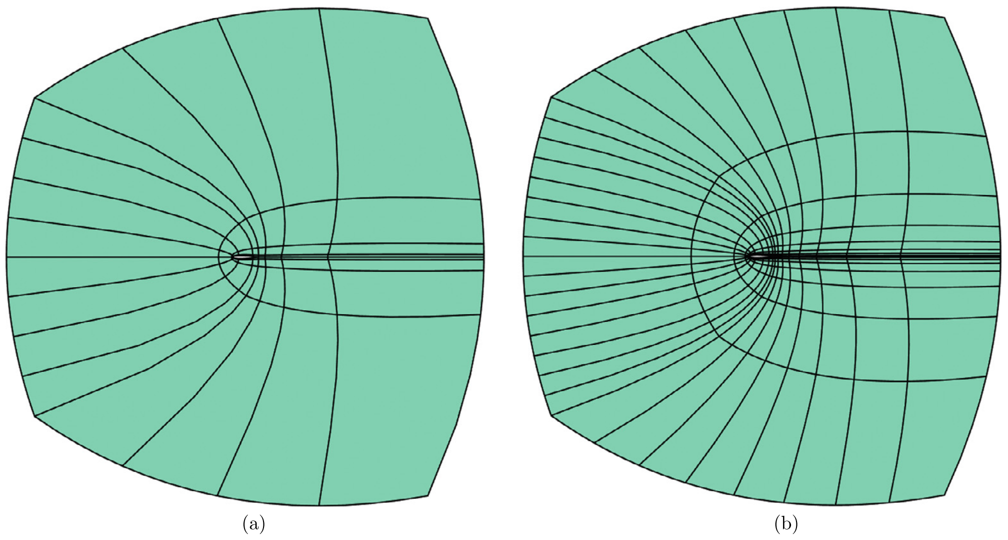

Fig. 7. Viscous subsonic flow past a NACA0012 airfoil: (a) the coarsest grid of 4×16 elements and (b) its first refinement of 8×32 elements.

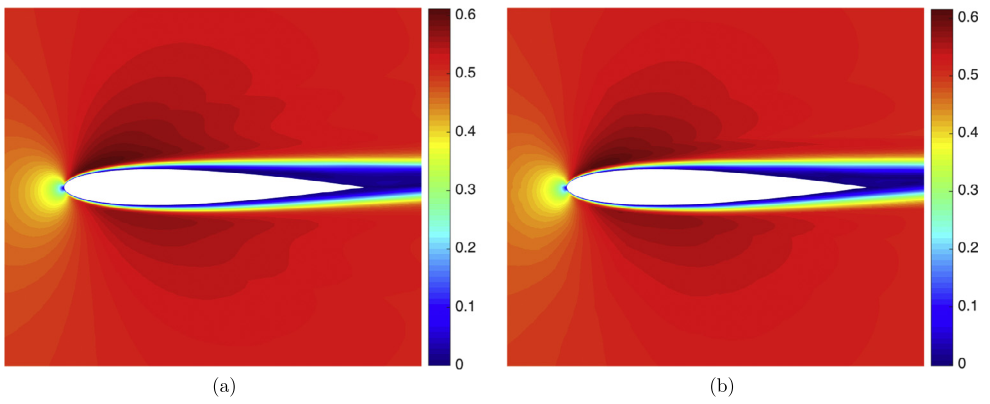

Fig. 8. Viscous subsonic flow past a NACA0012 airfoil: The Mach number of the numerical solutions computed using the IEDG method on (a) the coarsest mesh with $k = 4$ and (b) the first refinement with $k = 2$.

Table 10
Viscous subsonic flow past a NACA0012 airfoil: Errors in the lift coefficient and drag coefficient for the HDG, EDG, and IEDG solutions.

Degree k	Mesh $m \times n$	Error in lift coefficient			Error in drag coefficient		
		HDG	EDG	IEDG	HDG	EDG	IEDG
1	4×16	2.22×10^{-1}	3.15×10^{-1}	2.57×10^{-1}	2.93×10^{-2}	3.48×10^{-2}	3.11×10^{-2}
	8×32	6.59×10^{-2}	8.94×10^{-2}	6.65×10^{-2}	8.03×10^{-3}	9.68×10^{-3}	8.47×10^{-2}
	16×64	1.71×10^{-2}	2.26×10^{-2}	1.73×10^{-2}	2.43×10^{-3}	3.12×10^{-3}	2.76×10^{-3}
2	4×16	8.54×10^{-2}	9.05×10^{-2}	8.67×10^{-2}	8.01×10^{-3}	8.70×10^{-3}	8.32×10^{-3}
	8×32	1.32×10^{-2}	1.43×10^{-2}	1.35×10^{-2}	1.23×10^{-3}	1.38×10^{-3}	1.27×10^{-3}
	16×64	1.90×10^{-3}	1.95×10^{-3}	1.87×10^{-3}	1.64×10^{-4}	2.20×10^{-4}	1.98×10^{-4}
3	4×16	1.95×10^{-2}	1.94×10^{-2}	1.92×10^{-2}	1.72×10^{-3}	2.09×10^{-3}	1.88×10^{-3}
	8×32	1.32×10^{-3}	1.56×10^{-3}	1.39×10^{-3}	1.21×10^{-4}	1.52×10^{-4}	1.35×10^{-4}
	16×64	9.82×10^{-5}	1.08×10^{-4}	9.97×10^{-5}	8.75×10^{-6}	1.20×10^{-5}	9.09×10^{-6}
4	4×16	6.84×10^{-3}	7.38×10^{-3}	6.79×10^{-3}	6.37×10^{-4}	6.98×10^{-4}	6.52×10^{-4}
	8×32	2.71×10^{-4}	3.06×10^{-4}	2.75×10^{-4}	2.49×10^{-5}	2.82×10^{-5}	2.65×10^{-5}
	16×64	1.13×10^{-5}	1.35×10^{-5}	1.21×10^{-5}	9.71×10^{-7}	1.20×10^{-6}	1.05×10^{-6}

illustrates again the benefit of using polynomials of high degree to approximate the solution and geometry. Note that all the methods converge optimally with order $k + 1$ for $k = 1$, while they do not converge optimally for $k > 1$ due to singularity near the trailing edge. The IEDG method is just as accurate as the HDG method, while having significant less degrees of freedom.

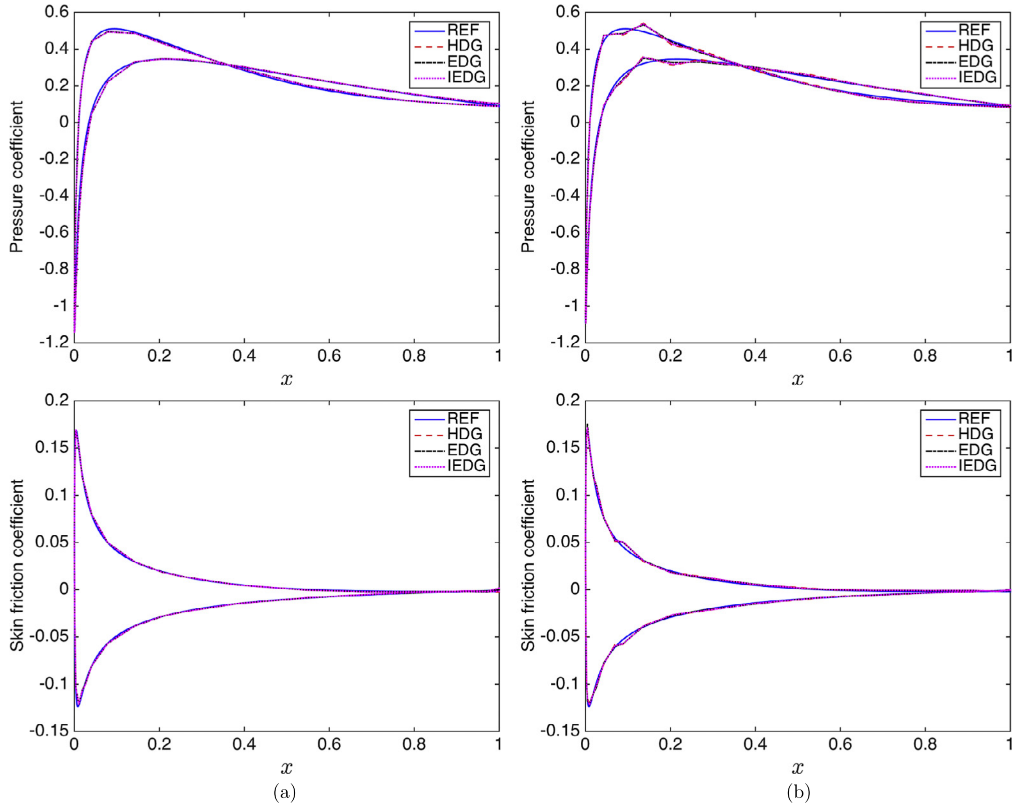

Fig. 9. Viscous subsonic flow past a NACA0012 airfoil: Pressure coefficient (top) and skin friction coefficient (bottom) for (a) the coarsest mesh and $k = 4$ and (b) the first refinement with $k = 2$. Note that REF stands for the reference solution.

5. Conclusions

We have presented a class of embedded discontinuous Galerkin methods for numerically solving systems of conservation laws arising in computational fluid dynamics. Although the paper is focused on the Euler and Navier–Stokes equations, the EDG method can be used to solve other systems of equations. The proposed EDG methods inherit many features of HDG methods and thus possess unique advantages which distinguish themselves from other DG methods. The main advantage of the EDG methods is that it gives rise to a matrix system which has the sparsity structure of the statically condensed CG method, while requiring less memory storage and computation time than other DG methods.

In this paper, we studied two particular EDG methods: the original EDG method presented in our previous work [55] is constructed by requiring the approximate trace to be continuous on all the faces, while the IEDG method is constructed by enforcing the continuity of the approximate trace only on the interior faces. There are two advantages of the IEDG method relative to the original EDG method. First, with the IEDG method, the boundary conditions are enforced more rigorously and accurately because the local spaces for the approximate trace on the domain boundary are exactly the same as those in the HDG method. Second, for the IEDG method, the degrees of freedom of the approximate trace on the domain boundary can be locally condensed without changing the sparsity pattern of the Jacobian matrix. Numerical results presented herein show that the accuracy of the IEDG method appears to be comparable to that of the HDG method for many test cases. This feature makes the IEDG method computationally competitive with other finite element methods and finite volume methods.

Both the HDG and EDG methods have been used to solve RANS flows, shock flows, as well as unsteady flows [44–46,55,56]. Future work will focus on extending the IEDG method to simulate these flows and developing effective Newton–Krylov solvers for solving the algebraic systems arising from the IEDG discretization of nonlinear systems of conservation laws in fluid mechanics. We close the paper by noting that the EDG methods can be combined with the discontinuous Petrov–Galerkin (DPG) method [30] to capture shocks more robustly. This line of research has been investigated in our recent work [42,43], wherein the DPG method is used for solving the local problems of the HDG method. The resulting hybridized DPG scheme enables shock capturing of the viscous Burgers equation. The EDG methods may also benefit from the DPG method just as the same way as the HDG method does. This is a subject worthy of further investigation.

References

[1] R. Alexander, Diagonally implicit Runge–Kutta methods for stiff ODEs, SIAM J. Numer. Anal. 14 (1977) 1006–1021.

[2] D.A. Anderson, J.C. Tannehill, R.H. Pletcher, Computational Fluid Dynamics and Heat Transfer, Hemisphere Publishing, New York, 1984.
[3] D.N. Arnold, F. Brezzi, B. Cockburn, L.D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (5) (2001) 1749–1779.
[4] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, J. Comput. Phys. 131 (2) (1997) 267–279.
[5] C. Baumann, J. Oden, A discontinuous hp finite element method for convection–diffusion problems, Comput. Methods Appl. Mech. Eng. 175 (1999) 311–341.
[6] F. Brezzi, J. Douglas Jr., L.D. Marini, Two families of mixed finite elements for second order elliptic problems, Numer. Math. 47 (1985) 217–235.
[7] A. Buffa, T.J.R. Hughes, G. Sangalli, Analysis of a multiscale discontinuous Galerkin method for convection–diffusion problems, SIAM J. Numer. Anal. 44 (2006) 1420–1440.
[8] Y. Chen, B. Cockburn, Analysis of variable-degree HDG methods for convection–diffusion equations. Part I: General nonconforming meshes, IMA J. Numer. Anal. 32 (2012) 1267–1293.
[9] Y. Chen, B. Cockburn, Analysis of variable-degree HDG methods for convection–diffusion equations. Part II: Semimatching nonconforming meshes, Math. Comput. 83 (2014) 87–111.
[10] G. Chiocchia, Exact solutions to transonic and supersonic flows, Technical Report AR-211, AGARD, 1985.
[11] B. Cockburn, B. Dong, J. Guzmán, A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems, Math. Comput. 77 (2008) 1887–1916.
[12] B. Cockburn, B. Dong, J. Guzmán, M. Restelli, R. Sacco, A hybridizable discontinuous Galerkin method for steady-state convection–diffusion–reaction problems, SIAM J. Sci. Comput. 31 (5) (2009) 3827–3846.
[13] B. Cockburn, J. Gopalakrishnan, The derivation of hybridizable discontinuous Galerkin methods for Stokes flow, SIAM J. Numer. Anal. 47 (2009) 1092–1125.
[14] B. Cockburn, J. Gopalakrishnan, R. Lazarov, Unified hybridization of discontinuous Galerkin, mixed and continuous Galerkin methods for second order elliptic problems, SIAM J. Numer. Anal. 47 (2009) 1319–1365.
[15] B. Cockburn, J. Gopalakrishnan, N.C. Nguyen, J. Peraire, F.-J. Sayas, Analysis of HDG methods for Stokes flow, Math. Comput. 80 (2011) 723–760.
[16] B. Cockburn, J. Gopalakrishnan, F.-J. Sayas, A projection-based error analysis of HDG methods, Math. Comput. 79 (2010) 1351–1367.
[17] B. Cockburn, J. Guzmán, S.-C. Soon, H.K. Stolarski, An analysis of the embedded discontinuous Galerkin method for second-order elliptic problems, SIAM J. Numer. Anal. 47 (2009) 2686–2707.
[18] B. Cockburn, J. Guzmán, H. Wang, Superconvergent discontinuous Galerkin methods for second-order elliptic problems, Math. Comput. 78 (2009) 1–24.
[19] B. Cockburn, G. Kanschat, D. Schötzau, C. Schwab, Local discontinuous Galerkin methods for the Stokes system, SIAM J. Numer. Anal. 40 (2002) 319–343.
[20] B. Cockburn, G. Kanschat, D. Schötzau, Local discontinuous Galerkin methods for the Oseen equations, Math. Comput. 73 (2004) 569–593.
[21] B. Cockburn, G. Kanschat, D. Schötzau, A locally conservative LDG method for the incompressible Navier–Stokes equations, Math. Comput. 74 (2005) 1067–1095.
[22] B. Cockburn, G. Kanschat, D. Schötzau, A note on discontinuous Galerkin divergence-free solutions of the Navier–Stokes equations, J. Sci. Comput. 31 (2007) 61–73.
[23] B. Cockburn, G. Kanschat, D. Schötzau, An equal-order DG method for the incompressible Navier–Stokes equations, J. Sci. Comput. 40 (2009) 141–187.
[24] B. Cockburn, W. Qiu, K. Shi, Conditions for superconvergence of HDG methods for second-order elliptic problems, Math. Comput. 81 (2012) 1327–1353.
[25] B. Cockburn, W. Qui, K. Shi, Conditions for superconvergence of HDG methods on curvilinear elements for second-order elliptic problems, SIAM J. Numer. Anal. 50 (2012) 1417–1432.
[26] B. Cockburn, K. Shi, Conditions for superconvergence of HDG methods for Stokes flow, Math. Comput. 82 (2013) 651–671.
[27] B. Cockburn, K. Shi, Superconvergent HDG methods for linear elasticity with weakly symmetric stresses, IMA J. Numer. Anal. 33 (2013) 747–770.
[28] B. Cockburn, C.W. Shu, The local discontinuous Galerkin method for convection–diffusion systems, SIAM J. Numer. Anal. 35 (1998) 2440–2463.
[29] B. Cockburn, C.-W. Shu, Runge–Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (3) (2001) 173–261.
[30] L. Demkowicz, J. Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. Part I: The transport equation, Comput. Methods Appl. Mech. Eng. 199 (2010) 1558–1572.
[31] K.J. Fidkowski, T.A. Oliver, J. Lu, D.L. Darmofal, *p*-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, J. Comput. Phys. 207 (1) (2005) 92–113.
[32] S. Güzey, B. Cockburn, H. Stolarski, The embedded discontinuous Galerkin methods: application to linear shell problems, Int. J. Numer. Methods Eng. 70 (2007) 757–790.
[33] R.M. Kirby, S.J. Sherwin, B. Cockburn, To HDG or to CG: a comparative study, J. Sci. Comput. 51 (2012) 183–212.
[34] R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, J. Comput. Phys. 183 (2002) 508–532.
[35] J.S. Hesthaven, T. Warburton, Nodal high-order methods on unstructured grids, I. Time-domain solution of Maxwell's equations, J. Comput. Phys. 181 (1) (2002) 186–221.
[36] A. Huerta, A. Angeloski, X. Roca, J. Peraire, Efficiency of high-order elements for continuous and discontinuous Galerkin methods, Int. J. Numer. Methods Eng. 96 (2013) 529–560.
[37] T.J.R. Hughes, G. Scovazzi, P.B. Bochev, A. Buffa, A multiscale discontinuous Galerkin method with the computational structure of a continuous Galerkin method, Comput. Methods Appl. Mech. Eng. 195 (2006) 2761–2787.
[38] C.M. Klaij, J.J.W. van der Vegt, H. van der Ven, Space–time discontinuous Galerkin method for the compressible Navier–Stokes equations, J. Comput. Phys. 217 (2) (2006) 589–611.
[39] I. Lomtev, G.E. Karniadakis, A discontinuous Galerkin method for the Navier–Stokes equations, Int. J. Numer. Methods Fluids 29 (1999) 587–603.
[40] I. Lomtev, R.M. Kirby, G.E. Karniadakis, A discontinuous Galerkin ALE method for compressible viscous flows in moving domains, J. Comput. Phys. 155 (1) (1999) 128–159.
[41] A. Montlaur, S. Fernández-Méndez, J. Peraire, A. Huerta, Discontinuous Galerkin methods for the Navier–Stokes equations using solenoidal approximations, Int. J. Numer. Methods Fluids 64 (2010) 549–564.
[42] D. Moro, N.C. Nguyen, J. Peraire, J. Gopalakrishnan, A hybridized discontinuous Petrov–Galerkin method for compressible flows (AIAA Paper 2011-197), in: Proceedings of the 49th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, FL, January 2011.
[43] D. Moro, N.C. Nguyen, J. Peraire, A hybridized discontinuous Petrov–Galerkin scheme for scalar conservation laws, Int. J. Numer. Methods Eng. 91 (2012) 950–970.
[44] D. Moro, N.C. Nguyen, J. Peraire, Navier–Stokes solution using hybridizable discontinuous Galerkin methods (AIAA Paper 2011-3060), in: Proceedings of the 20th AIAA Computational Fluid Dynamics Conference, Honolulu, HI, June 2011.
[45] N.C. Nguyen, J. Peraire, Hybridizable discontinuous Galerkin methods for partial differential equations in continuum mechanics, J. Comput. Phys. 231 (2012) 5955–5988.
[46] N.C. Nguyen, J. Peraire, An adaptive shock-capturing HDG method for compressible flows (AIAA Paper 2011-3407), in: Proceedings of the 20th AIAA Computational Fluid Dynamics Conference, Honolulu, HI, June 2011.

[47] N.C. Nguyen, J. Peraire, B. Cockburn, A comparison of HDG methods for Stokes flow, J. Sci. Comput. 45 (1–3) (2010) 215–237.
[48] N.C. Nguyen, J. Peraire, B. Cockburn, Implicit high-order hybridizable discontinuous Galerkin methods for acoustics and elastodynamics, J. Comput. Phys. 230 (2011) 3695–3718.
[49] N.C. Nguyen, J. Peraire, B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier–Stokes equations, J. Comput. Phys. 230 (2011) 1147–1170.
[50] N.C. Nguyen, J. Peraire, B. Cockburn, Hybridizable discontinuous Galerkin methods, in: J.S. Hesthaven, E.M. Ronquist (Eds.), Spectral and High Order Methods for Partial Differential Equations, in: Lect. Notes Comput. Sci. Eng., vol. 76, 2011, pp. 63–84.
[51] N.C. Nguyen, J. Peraire, B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for linear convection–diffusion equations, J. Comput. Phys. 228 (2009) 3232–3254.
[52] N.C. Nguyen, J. Peraire, B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for nonlinear convection–diffusion equations, J. Comput. Phys. 228 (2009) 8841–8855.
[53] N.C. Nguyen, J. Peraire, B. Cockburn, A hybridizable discontinuous Galerkin method for Stokes flow, Comput. Methods Appl. Mech. Eng. 199 (2010) 582–597.
[54] N.C. Nguyen, J. Peraire, B. Cockburn, A hybridizable discontinuous Galerkin method for the incompressible Navier–Stokes equations (AIAA Paper 2010-362), in: Proceedings of the 48th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, FL, January 2010.
[55] J. Peraire, N.C. Nguyen, B. Cockburn, An embedded discontinuous Galerkin method for the compressible Euler and Navier–Stokes equations (AIAA Paper 2011-3228), in: Proceedings of the 20th AIAA Computational Fluid Dynamics Conference, Honolulu, HI, 2011.
[56] J. Peraire, N.C. Nguyen, B. Cockburn, A hybridizable discontinuous Galerkin method for the compressible Euler and Navier–Stokes equations (AIAA Paper 2010-363), in: Proceedings of the 48th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, FL, January 2010.
[57] J. Peraire, P.O. Persson, The compact discontinuous Galerkin (CDG) method for elliptic problems, SIAM J. Sci. Comput. 30 (4) (2008) 1806–1824.
[58] P.O. Persson, J. Peraire, Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier–Stokes equations, SIAM J. Sci. Comput. 30 (6) (2008) 2709–2733.
[59] P.-O. Persson, J. Bonet, J. Peraire, Discontinuous Galerkin solution of the Navier–Stokes equations on deformable domains, Comput. Methods Appl. Mech. Eng. 198 (2009) 1585–1595.
[60] P.A. Raviart, J.M. Thomas, A mixed finite element method for second order elliptic problems, in: I. Galligani, E. Magenes (Eds.), Mathematical Aspects of Finite Element Method, in: Lect. Notes Math., vol. 606, Springer-Verlag, New York, 1977, pp. 292–315.
[61] W.H. Reed, T.R. Hill, Triangular mesh methods for the neutron transport equation, Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.
[62] X. Roca, J. Peraire, N.C. Nguyen, Scalable parallelization of the hybridized discontinuous Galerkin method for compressible flow (AIAA Paper 2013–2939), in: Proceedings of the 21th AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 2013.
[63] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (1981) 357–372.
[64] K. Shahbazi, P.F. Fischer, C.R. Ethier, A high-order discontinuous Galerkin method for the unsteady incompressible Navier–Stokes equations, J. Comput. Phys. 222 (2007) 391–407.
[65] M.P. Ueckermann, P.F.J. Lermusiaux, High order schemes for 2D unsteady biogeochemical ocean models, Ocean Dyn. 60 (2010) 1415–1445.
[66] Z.J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H.T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, M. Visbal, High-order CFD methods: current status and perspective, Int. J. Numer. Methods Fluids 72 (2013) 811–845.