Fluid Dynamics and Co-located Conferences
June 24-27, 2013, San Diego, CA
21st AIAA Computational Fluid Dynamics Conference

10.2514/6.2013-2939

Scalable parallelization of the hybridized discontinuous Galerkin method for compressible flow

Xevi Roca*, Ngoc C. Nguyen† and Jaime Peraire‡
Massachusetts Institute of Technology, Cambridge, MA, 02139, USA

A distributed and parallel implementation of a hybridized discontinuous Galerkin (HDG) solver for the compressible Navier-Stokes equations is presented. This implementation exploits the characteristics of the HDG method. First, the global degrees of freedom are reduced to the numerical trace of the solution on the element boundaries. Second, the conserved quantities and their gradients on each element are obtained in terms of the numerical trace on the element boundary. For meshes composed by a large numer of elements, the cost of the solver is dominated by the first stage, since the second stage scales linearly with the number of elements and it can be parallelized. To accelerate the first stage, we use a distributed GMRES solver with restart pre-conditioned with an algebraic additive Schwarz domain decomposition with l-levels of overlap (ASDD(l)). Each sub-domain problem is approximated by an incomplete LU factorization with k-levels of fill-in (ILU(k)). From the considered tests cases, we conclude that ASDD(1)/ILU(0) presents good weak scaling characteristics for studying the periodic vortex shedding around an airfoil at low Reynolds and low Mach number.

I. Introduction

The conservative and stabilized formulation of the discontinuous Galerkin methods provides some benefits when it is applied to solve conservation problems.[1,2] Recently, the hybridized discontinuous Galerkin method has been introduced in Ref. [3] and further developed in Refs. [4–13]. Several unique features distinguish the HDG methods from other discontinuous Galerkin methods. First, they provide, for smooth and viscous-dominated problems, approximations of all the variables which converge with the optimal order of $k+1$ in the L^2-norm. Second, they possess some superconvergence properties that allow to compute a new approximate displacement and velocity which converge with order $k+2$ for $k \geq 1$ by means of an inexpensive element-by-element postprocessing procedure. Third, they provide a novel and systematic way for imposing several types of boundary conditions. Fourth, the linear systems that arise from the HDG method is equivalent to two different linear systems: a first one that couples globally the numerical trace of the solution on element boundaries, thereby leading to a significant reduction in the degrees of freedom; and a second one that couples at the element level (locally) the conserved quantities and their gradients and therefore, can be solved in an element-by-element manner.

The properties above arise from the three main ingredients of the HDG method: (1) a local Galerkin projection of the underlying PDEs at the element level onto suitable approximation spaces of order k is performed to parametrize the approximate the solution in terms of the numerical trace; (2) a global jump condition that enforces the continuity of the numerical flux is stated to derive a global weak formulation in terms of the numerical trace; and (3) a judicious choice of the numerical flux allows to provide stability and consistency.

It is important to point out that ingredients (1) and (2) lead to the possibility of solving the linear system in two stages. This is the advantage that we exploit to propose a scalable parallel implementation of the HDG method for solving the compressible Navier-Stokes equations. First, we solve the global linear system with a parallel GMRES iterative solver pre-conditioned with an additive Schwarz domain-decomposition

*Post-doctoral Associate, Dep. of Aeronautics and Astronautics, MIT, 77 Massachusetts Avenue, AIAA Member.
†Research Scientist, Dep. of Aeronautics and Astronautics, MIT, 77 Massachusetts Avenue, AIAA Member.
‡Professor and Head, Dep. of Aeronautics and Astronautics, MIT, 77 Massachusetts Avenue, AIAA Fellow.

with l-levels of overlap.[14] Moreover, each one of the sub-problems is approximated using an incomplete ILU factorization with k-levels of fill-in. Second, we solve in parallel the local problems at the element level with small amount of communication. The performance of the HDG method is dominated by the solution of the global linear system, since the local problem scales linearly with the number of elements and is parallelized.

The proposed parallel algorithm presents some advantages when it is compared with a naïve parallel implementation. That is, such method would solve a monolithic linear system expressed in terms of all the variables: the conserved quantities, and their gradients at the element level, and the conserved quantities at the element boundaries. On the contrary, here we propose to solve in parallel the local problems (element level) with strong scaling properties, and a reduced global system (face level) that requires communication but with less degrees of freedom. Note that iterative solvers on the CPU have been developed for the CDG method[15] as well as other implicit DG methods[1,16–18] . In the case of the HDG method, a deep study of domain decomposition methods is presented in Ref. 19, and a GPU-accelerated sparse matrix-vector product for iterative solvers has been proposed in Ref. 20.

The remainder of this paper is organized as follows. In Section II, we present the HDG method and its application to the compressible Navier-Stokes equations. In Section III, we detail the Newton iteration for the weak formulation of the HDG method and how to implement it in a sequential processor. In Section IV, we exploit the structure of the linear systems that arise from the HDG method to propose a distributed and parallel implementation. In Section V, we present the obtained results.

II. HDG method for compressible flow

In this work, we want to solve the compressible Navier-Stokes equations using the HDG method. The general weak formulation of the HDG method allows solving second-order partial differential equations resulting from conservative laws, see Section II.A. By choosing the proper values of the fluxes, boundary operators, boundary functions, and the source term, the general weak formulation can be particularized to solve the compressible flow equations, see Section II.B.

II.A. The HDG method

The HDG method allows solving second-order systems of partial differential equations for conservation laws, see Section II.A.1. To derive the HDG method, an equivalent first-order problem is obtained by introducing a new variable that represents the gradient of the conserved quantities, see Section II.A.2. To obtain the weak formulation of the method, it is required to introduce the proper discontinuous spaces on the mesh elements and faces, see Section II.A.3. Then, the general weak form of the method is derived in Section II.A.4.

II.A.1. Second-order problems for conservation laws

In general terms, the HDG method allows solving second-order partial differential equations in conservative form:

$$\begin{aligned} \alpha \boldsymbol{u} + \nabla \cdot \boldsymbol{F}(\boldsymbol{u}, \nabla \boldsymbol{u}) &= s, && \text{in } \Omega \subset \mathbb{R}^d, \\ \boldsymbol{u} &= g_D, && \text{on } \Gamma_D, \\ \boldsymbol{B}(\boldsymbol{u}, \nabla \boldsymbol{u}) \cdot \boldsymbol{n} &= g_N, && \text{on } \Gamma_N, \end{aligned} \tag{1}$$

where α is a scalar, $\boldsymbol{F}$ is a $(n_c \times d)$-dimensional flux, s a source term, and $\boldsymbol{B}$ an operator that determines the boundary conditions.

II.A.2. Equivalent first-order problem

By introducing a new variable $\boldsymbol{q} = \nabla \boldsymbol{u}$, we obtain an equivalent first-order problem:

$$\begin{aligned} \boldsymbol{q} - \nabla \boldsymbol{u} &= \boldsymbol{0}, && \text{in } \Omega \subset \mathbb{R}^d, \\ \alpha \boldsymbol{u} + \nabla \cdot \boldsymbol{F}(\boldsymbol{u}, \boldsymbol{q}) &= s, && \text{in } \Omega \subset \mathbb{R}^d, \\ \boldsymbol{u} &= g_D, && \text{on } \Gamma_D, \\ \boldsymbol{B}(\boldsymbol{u}, \boldsymbol{q}) \cdot \boldsymbol{n} &= g_N, && \text{on } \Gamma_N. \end{aligned} \tag{2}$$

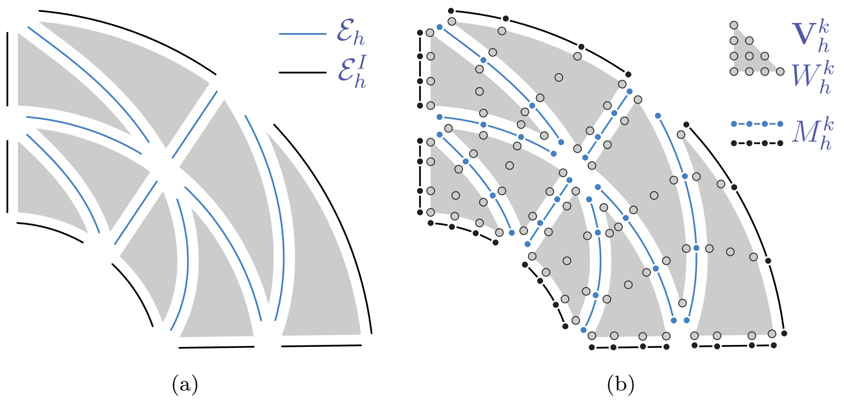


Figure 1. HDG discretization: (a) curved and discontinuous elements and faces; and (b) nodes of the corresponding approximation spaces.

II.A.3. Notation

To describe the HDG weak formulation, we introduce some notation. First, the HDG mesh for a domain Ω (*e.g.* Figure 1(a)) is composed by discontinuous and potentially curved elements ($\mathcal{T}_h$) and faces ($\mathcal{E}_h$). The faces can be classifed as interior faces ($\mathcal{E}_h^I$) and boundary faces ($\mathcal{E}_h^B$).

Second, let be $\mathcal{P}_k(D)$ denote the space of polynomials of degree at most k on a domain D and let $L^2(D)$ be the space of square integrable functions on D. We introduce the following discontinuous finite element approximation spaces:

$$\mathcal{W}_h^k = \{w \in L^2(\mathcal{T}_h) \mid w_{|_K} \in \mathcal{P}_k(K), \forall K \in \mathcal{T}_h\},$$
$$\boldsymbol{\mathcal{W}}_h^k = \{\boldsymbol{w} \in (L^2(\mathcal{T}_h))^{n_c} \mid \boldsymbol{w}_{|_K} \in (\mathcal{P}_k(K))^{n_c}, \forall K \in \mathcal{T}_h\},$$
$$\boldsymbol{\mathcal{V}}_h^k = \{\boldsymbol{v} \in (L^2(\mathcal{T}_h))^{n_c \times d} \mid \boldsymbol{v}_{|_K} \in [\mathcal{P}_k(K)]^{n_c \times d}, \forall K \in \mathcal{T}_h\}.$$

Moreover, we introduce the following finite element spaces for the approximate trace of the solution:

$$\mathcal{M}_h^k = \{\mu \in L^2(\mathcal{E}_h) \mid \mu_{|_F} \in \mathcal{P}_k(F), \forall F \in \mathcal{E}_h\},$$
$$\boldsymbol{\mathcal{M}}_h^k = \{\boldsymbol{\mu} \in (L^2(\mathcal{E}_h))^{n_c} \mid \boldsymbol{\mu}_{|_F} \in (\mathcal{P}_k(F))^{n_c}, \forall F \in \mathcal{E}_h\}.$$

Note that $\mathcal{M}_h^k$ and $\boldsymbol{\mathcal{M}}_h^k$ both consist of functions which are continuous inside the faces $F \in \mathcal{E}_h$ and discontinuous at their borders. To illustrate the introduced approximation spaces, we present in Figure 1(b) the corresponding element and face nodes for a curved mesh of interpolation degree $k = 3$.

Finally, we define the corresponding inner products for functions in the element spaces:

$$\left(w^1, w^2\right)_{\mathcal{T}_h} := \sum_{K \in \mathcal{T}_h} \left(w^1, w^2\right)_K, \text{ for } w^1, w^2 \in \mathcal{W}_h^k,$$
$$\left(\boldsymbol{w}^1, \boldsymbol{w}^2\right)_{\mathcal{T}_h} := \sum_{i=1}^{m} \left(\boldsymbol{w}_i^1, \boldsymbol{w}_i^2\right)_{\mathcal{T}_h}, \text{ for } \boldsymbol{w}^1, \boldsymbol{w}^2 \in \boldsymbol{\mathcal{W}}_h^k,$$
$$\left(\boldsymbol{v}^1, \boldsymbol{v}^2\right)_{\mathcal{T}_h} := \sum_{i=1}^{m} \sum_{j=1}^{d} \left(\boldsymbol{v}_{ij}^1, \boldsymbol{v}_{ij}^2\right)_{\mathcal{T}_h}, \text{ for } \boldsymbol{v}^1, \boldsymbol{v}^2 \in \boldsymbol{\mathcal{V}}_h^k,$$

where $\left(w^1, w^2\right)_K := \int_K w^1 w^2$. In addition, the corresponding inner product for functions in the face spaces:

$$\left\langle \mu^1, \mu^2 \right\rangle_{\partial \mathcal{T}_h} := \sum_{K \in \mathcal{T}_h} \left\langle \mu^1, \mu^2 \right\rangle_{\partial K}, \text{ for } \mu^1, \mu^2 \in \mathcal{M}_h^k,$$
$$\left\langle \boldsymbol{\mu}^1, \boldsymbol{\mu}^2 \right\rangle_{\partial \mathcal{T}_h} := \sum_{i=1}^{m} \left\langle \boldsymbol{\mu}^1, \boldsymbol{\mu}^2 \right\rangle_{\partial \mathcal{T}_h}, \text{ for } \boldsymbol{\mu}^1, \boldsymbol{\mu}^2 \in \boldsymbol{\mathcal{M}}_h^k,$$

where $\left\langle \mu^1, \mu^2 \right\rangle_{\partial K} := \int_{\partial K} \mu^1 \mu^2$.

II.A.4. Weak formulation

The HDG method seeks a solution $(\boldsymbol{q}_h, \boldsymbol{u}_h, \hat{\boldsymbol{u}}_h) \in \boldsymbol{\mathcal{V}}_h^k \times \boldsymbol{\mathcal{W}}_h^k \times \boldsymbol{\mathcal{M}}_h^k$ such that:

$$\mathbf{r_q} := (\boldsymbol{q}_h, \boldsymbol{v})_{\mathcal{T}_h} + (\boldsymbol{u}_h, \nabla \cdot \boldsymbol{v})_{\mathcal{T}_h} - \langle \hat{\boldsymbol{u}}_h, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial \mathcal{T}_h} = 0, \ \forall \boldsymbol{v} \in \boldsymbol{\mathcal{V}}_h^k, \tag{3}$$

$$\mathbf{r_u} := \alpha \left(\boldsymbol{u}_h, \boldsymbol{w}\right)_{\mathcal{T}_h} - (\boldsymbol{F}, \nabla \boldsymbol{w})_{\mathcal{T}_h} + \left\langle \hat{\boldsymbol{F}} \cdot \boldsymbol{n}, \boldsymbol{w} \right\rangle_{\partial \mathcal{T}_h} - (s, \boldsymbol{w})_{\mathcal{T}_h} = 0, \ \forall \boldsymbol{w} \in \boldsymbol{\mathcal{W}}_h^k, \tag{4}$$

$$\mathbf{r_{\hat{u}}} := \left\langle \hat{\boldsymbol{F}} \cdot \boldsymbol{n}, \boldsymbol{\mu} \right\rangle_{\partial \mathcal{T}_h \setminus \partial\Omega} + \left\langle \hat{\boldsymbol{F}}^b \cdot \boldsymbol{n}, \boldsymbol{\mu} \right\rangle_{\partial\Omega} - \langle g, \boldsymbol{\mu} \rangle_{\partial\Omega} = 0, \ \forall \boldsymbol{\mu} \in \boldsymbol{\mathcal{M}}_h^k, \tag{5}$$

where $\hat{\boldsymbol{F}} = \hat{\boldsymbol{F}}(\boldsymbol{q}_h, \boldsymbol{u}_h, \hat{\boldsymbol{u}}_h)$ is the numerical flux (stabilization), and $\hat{\boldsymbol{F}}^b = \hat{\boldsymbol{F}}^b(\boldsymbol{q}_h, \boldsymbol{u}_h, \hat{\boldsymbol{u}}_h)$ determines the appropiate boundary conditions on Γ_D and Γ_N. Specifically, the numerical flux is defined as

$$\hat{\boldsymbol{F}}(\boldsymbol{q}_h, \boldsymbol{u}_h, \hat{\boldsymbol{u}}_h) \cdot \boldsymbol{n} := \boldsymbol{F}(\hat{\boldsymbol{u}}_h, \boldsymbol{q}_h) \cdot \boldsymbol{n} - \mathbf{S}(\boldsymbol{u}_h, \hat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \hat{\boldsymbol{u}}_h),$$

where $\mathbf{S}(\boldsymbol{u}_h, \hat{\boldsymbol{u}}_h)$ is the *stabilization matrix.* In addition, the boundary conditions are determined by

$$\hat{\boldsymbol{F}}^b \cdot \boldsymbol{n} := \begin{cases} \hat{\boldsymbol{u}}_h, & \text{on } \Gamma_D, \\ \boldsymbol{B}(\hat{\boldsymbol{u}}_h, \boldsymbol{q}_h) \cdot \boldsymbol{n} - \mathbf{S}(\boldsymbol{u}_h, \hat{\boldsymbol{u}}_h)(\boldsymbol{u}_h - \hat{\boldsymbol{u}}_h), & \text{on } \Gamma_N, \end{cases}$$

where

$$g := \begin{cases} g_D, & \text{on } \Gamma_D, \\ g_N, & \text{on } \Gamma_N. \end{cases}$$

On the one hand, Equations (3) and (4) allow to parameterize $(\boldsymbol{q}_h, \boldsymbol{u}_h)$ in terms of $\hat{\boldsymbol{u}}_h$ element-by-element (locally). On the other hand, Equation (5) is responsible of imposing the continuity of $\hat{\boldsymbol{F}}$ (globally) and $\hat{\boldsymbol{F}}^b$ (boundary conditions). Note that in $\hat{\boldsymbol{F}}$ and $\hat{\boldsymbol{F}}^b$, we $\hat{\boldsymbol{u}}_h$ instead of $\boldsymbol{u}_h$ in the evaluation of both $\boldsymbol{F}$ and $\boldsymbol{B}$.

II.B. HDG method for compressible flow

We want to solve the steady-state Navier-Stokes equations written in conservative form as:

$$\begin{aligned} \nabla \cdot (\boldsymbol{F}_{\text{inv.}}(\boldsymbol{u}) + \boldsymbol{F}_{\text{vis.}}(\boldsymbol{u}, \nabla \boldsymbol{u})) &= s, && \text{in } \Omega \subset \mathbb{R}^d, \\ \boldsymbol{u} &= g_D, && \text{on } \Gamma_D, \\ \boldsymbol{B}(\boldsymbol{u}, \nabla \boldsymbol{u}) \cdot \boldsymbol{n} &= g_N, && \text{on } \Gamma_N, \end{aligned} \tag{6}$$

where: $\boldsymbol{u}$ are the conserved quantities (density, momentum, and energy); $\boldsymbol{F}_{\text{inv.}}(\boldsymbol{u})$ and $\boldsymbol{F}_{\text{vis.}}(\boldsymbol{u}, \nabla \boldsymbol{u})$ are the $((d+2) \times d)$-dimensional inviscid and viscous fluxes, respectively; s is the source term; and $\boldsymbol{B}$ and g are defined to determine the appropiate boundary conditions at the inflow, outflow, ans solid wall boundaries. The nondimensional form of the Navier-Stokes equations as well as the definition of the inviscid and viscous fluxes can be found in.[21] Note that this conservative form corresponds to the general form in Equation (1) for $\alpha = 0$, $\boldsymbol{F} = \boldsymbol{F}_{\text{inv.}} + \boldsymbol{F}_{\text{vis.}}$, and $n_c = d + 2$. Therefore, the HDG weak formulation for the compressible Navier-Stokes equations is obtained by substituting these particular expressions in the general weak form presented in Equations (3), (4), and (5).

III. HDG method: sequential implementation

The weak form of the HDG can be solved using the Newton method. The resulting linear systems are equivalent to solve a global linear system with a reduced number of degrees of freedom, and a set of local problems at the element level, see Section III.A. A sequential implementation that exploits these characteristics is presented in Section III.B

III.A. Linearization of the weak formulation

To solve the HDG weak formulation, we use the Newton method. Therefore, we have to linearize the components of the residual in Equations (3), (4), and (5) at each Newton iteration. This linearization leads to solve linear systems of the form:

$$\left[\begin{array}{c|c|c} \mathbf{J}_{\mathbf{qq}} & \mathbf{J}_{\mathbf{qu}} & \mathbf{J}_{\mathbf{q\hat{u}}} \\ \hline \mathbf{J}_{\mathbf{uq}} & \mathbf{J}_{\mathbf{uu}} & \mathbf{J}_{\mathbf{u\hat{u}}} \\ \hline \mathbf{J}_{\mathbf{\hat{u}q}} & \mathbf{J}_{\mathbf{\hat{u}u}} & \mathbf{J}_{\mathbf{\hat{u}\hat{u}}} \end{array}\right] \left[\begin{array}{c} \delta\mathbf{q} \\ \hline \delta\mathbf{u} \\ \hline \delta\mathbf{\hat{u}} \end{array}\right] = \left[\begin{array}{c} -\mathbf{r}_{\mathbf{q}} \\ \hline -\mathbf{r}_{\mathbf{u}} \\ \hline -\mathbf{r}_{\mathbf{\hat{u}}} \end{array}\right], \tag{7}$$

where $\delta\mathbf{q}$, $\delta\mathbf{u}$, and $\delta\hat{\mathbf{u}}$ are the degrees of freedom for $\boldsymbol{q}_h$, $\boldsymbol{u}_h$, and $\hat{\boldsymbol{u}}_h$, respectively. Taking into account Equation (7), we express $\delta\mathbf{q}$ and $\delta\mathbf{u}$ in terms of $\delta\hat{\mathbf{u}}$:

$$\left[\begin{array}{c} \delta\mathbf{q} \\ \hline \delta\mathbf{u} \end{array}\right] = \left[\begin{array}{c} \delta\mathbf{q}(\delta\hat{\mathbf{u}}) \\ \hline \delta\mathbf{u}(\delta\hat{\mathbf{u}}) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{J}_{\mathbf{qq}} & \mathbf{J}_{\mathbf{qu}} \\ \hline \mathbf{J}_{\mathbf{uq}} & \mathbf{J}_{\mathbf{uu}} \end{array}\right]^{-1} \left(\left[\begin{array}{c} -\mathbf{r}_{\mathbf{q}} \\ \hline -\mathbf{r}_{\mathbf{u}} \end{array}\right] - \left[\begin{array}{c} \mathbf{J}_{\mathbf{q\hat{u}}} \\ \hline \mathbf{J}_{\mathbf{u\hat{u}}} \end{array}\right] \delta\hat{\mathbf{u}} \right). \tag{8}$$

Finally, by eliminating $\delta\mathbf{q}$ and $\delta\mathbf{u}$ from Equation (7), we obtain the linear system:

$$\mathbf{H}\delta\hat{\mathbf{u}} = \mathbf{r}, \tag{9}$$

where

$$\mathbf{H} = \left[\begin{array}{c} \mathbf{J}_{\mathbf{\hat{u}\hat{u}}} \end{array}\right] - \left[\begin{array}{c|c} \mathbf{J}_{\mathbf{\hat{u}q}} & \mathbf{J}_{\mathbf{\hat{u}u}} \end{array}\right] \left[\begin{array}{c|c} \mathbf{J}_{\mathbf{qq}} & \mathbf{J}_{\mathbf{qu}} \\ \hline \mathbf{J}_{\mathbf{uq}} & \mathbf{J}_{\mathbf{uu}} \end{array}\right]^{-1} \left[\begin{array}{c} \mathbf{J}_{\mathbf{q\hat{u}}} \\ \hline \mathbf{J}_{\mathbf{u\hat{u}}} \end{array}\right], \tag{10}$$

$$\mathbf{r} = [-\mathbf{r}_{\mathbf{\hat{u}}}] - \left[\begin{array}{c|c} \mathbf{J}_{\mathbf{\hat{u}q}} & \mathbf{J}_{\mathbf{\hat{u}u}} \end{array}\right] \left[\begin{array}{c|c} \mathbf{J}_{\mathbf{qq}} & \mathbf{J}_{\mathbf{qu}} \\ \hline \mathbf{J}_{\mathbf{uq}} & \mathbf{J}_{\mathbf{uu}} \end{array}\right]^{-1} \left[\begin{array}{c} -\mathbf{r}_{\mathbf{q}} \\ \hline -\mathbf{r}_{\mathbf{u}} \end{array}\right]. \tag{11}$$

The system in Equation (9) contains only the degrees of freedom for $\delta\hat{\mathbf{u}}$.

III.B. Sequential solver

Note that at each iteration of the non-linear solver we need to assemble and solve the linear system in Equation (7). Taking into account the derivation above, we perform the following three steps: To obtain the expressions in Equations (9), (10), and (11), we need to compute the: elemental matrices $\mathbf{J}^K_{\mathbf{qq}}$, $\mathbf{J}^K_{\mathbf{qu}}$, $\mathbf{J}^K_{\mathbf{q\hat{u}}}$, $\mathbf{J}^K_{\mathbf{uq}}$, $\mathbf{J}^K_{\mathbf{uu}}$, $\mathbf{J}^K_{\mathbf{u\hat{u}}}$, $\mathbf{J}^K_{\mathbf{\hat{u}q}}$, $\mathbf{J}^K_{\mathbf{\hat{u}u}}$, and $\mathbf{J}^K_{\mathbf{\hat{u}\hat{u}}}$; and elemental vectors $\mathbf{r}^K_{\mathbf{q}}$, $\mathbf{r}^K_{\mathbf{u}}$, and $\mathbf{r}^K_{\mathbf{\hat{u}}}$. Furthermore, we have to compute the inverse of

$$\left[\begin{array}{c|c} \mathbf{J}_{\mathbf{qq}} & \mathbf{J}_{\mathbf{qu}} \\ \hline \mathbf{J}_{\mathbf{uq}} & \mathbf{J}_{\mathbf{uu}} \end{array}\right].$$

To this end, we group the degrees of freedom $\delta\mathbf{q}$ and $\delta\mathbf{u}$ by elements. Thus, the matrix becomes block diagonal and therefore, it can be inverted independently for each element K in $\mathcal{T}_h$:

$$\left[\begin{array}{c|c} \mathbf{J}^K_{\mathbf{qq}} & \mathbf{J}^K_{\mathbf{qu}} \\ \hline \mathbf{J}^K_{\mathbf{uq}} & \mathbf{J}^K_{\mathbf{uu}} \end{array}\right]^{-1}. \tag{12}$$

Furthermore, we can compute the elemental contribution to $\mathbf{H}$ and $\mathbf{r}$, Equations (10) and (11), for each element K in $\mathcal{T}_h$ as:

$$\mathbf{H}^K = \left[\begin{array}{c} \mathbf{J}^K_{\mathbf{\hat{u}\hat{u}}} \end{array}\right] - \left[\begin{array}{c|c} \mathbf{J}^K_{\mathbf{\hat{u}q}} & \mathbf{J}^K_{\mathbf{\hat{u}u}} \end{array}\right] \left[\begin{array}{c|c} \mathbf{J}^K_{\mathbf{qq}} & \mathbf{J}^K_{\mathbf{qu}} \\ \hline \mathbf{J}^K_{\mathbf{uq}} & \mathbf{J}^K_{\mathbf{uu}} \end{array}\right]^{-1} \left[\begin{array}{c} \mathbf{J}^K_{\mathbf{q\hat{u}}} \\ \hline \mathbf{J}^K_{\mathbf{u\hat{u}}} \end{array}\right], \tag{13}$$

$$\mathbf{r}^K = [-\mathbf{r}^K_{\mathbf{\hat{u}}}] - \left[\begin{array}{c|c} \mathbf{J}^K_{\mathbf{\hat{u}q}} & \mathbf{J}^K_{\mathbf{\hat{u}u}} \end{array}\right] \left[\begin{array}{c|c} \mathbf{J}^K_{\mathbf{qq}} & \mathbf{J}^K_{\mathbf{qu}} \\ \hline \mathbf{J}^K_{\mathbf{uq}} & \mathbf{J}^K_{\mathbf{uu}} \end{array}\right]^{-1} \left[\begin{array}{c} -\mathbf{r}^K_{\mathbf{q}} \\ \hline -\mathbf{r}^K_{\mathbf{u}} \end{array}\right]. \tag{14}$$

Finally, by assembling these element contributions we form the global linear system in Equation (9). We obtain $\delta\hat{\mathbf{u}}$ by solving the linear system in Equation (9).

To obtain $\delta\mathbf{q}$ and $\delta\mathbf{u}$ we substitute $\delta\hat{\mathbf{u}}$ in Equation (8). Moreover, we take into account that the degrees of freedom $\delta\mathbf{q}$ and $\delta\mathbf{u}$ are grouped by elements. That is, for each element K in $\mathcal{T}_h$ we compute $\delta\mathbf{q}^K$ and $\delta\mathbf{u}^K$ by means of:

$$\left[\begin{array}{c} \delta\mathbf{q}^K \\ \hline \delta\mathbf{u}^K \end{array}\right] = \left[\begin{array}{c} \delta\mathbf{q}(\delta\hat{\mathbf{u}}) \\ \hline \delta\mathbf{u}(\delta\hat{\mathbf{u}}) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{J}^K_{\mathbf{qq}} & \mathbf{J}^K_{\mathbf{qu}} \\ \hline \mathbf{J}^K_{\mathbf{uq}} & \mathbf{J}^K_{\mathbf{uu}} \end{array}\right]^{-1} \left(\left[\begin{array}{c} -\mathbf{r}^K_{\mathbf{q}} \\ \hline -\mathbf{r}^K_{\mathbf{u}} \end{array}\right] - \left[\begin{array}{c} \mathbf{J}^K_{\mathbf{q\hat{u}}} \\ \hline \mathbf{J}^K_{\mathbf{u\hat{u}}} \end{array}\right] \delta\hat{\mathbf{u}}^{\partial K} \right). \tag{15}$$

Algorithm 1: Sequential solver for the HDG linear system

```
    Input: $\mathbf{q}$, $\mathbf{u}$, $\hat{\mathbf{u}}$
    Output: $\delta\mathbf{q}$, $\delta\mathbf{u}$, $\delta\hat{\mathbf{u}}$
1   begin Compute matrix $\mathbf{H}$ and vector $\mathbf{r}$ from $\mathbf{q}$, $\mathbf{u}$, $\hat{\mathbf{u}}$
2       for $K$ in $\mathcal{T}_h$ do
3           $\mathbf{H}^K$, $\mathbf{r}^K \leftarrow$ Elemental contributions from $\mathbf{q}^K$, $\mathbf{u}^K$, $\hat{\mathbf{u}}^{\partial K}$
4       $\mathbf{H}$, $\mathbf{r} \leftarrow$ Assemble $\mathbf{H}^K$ and $\mathbf{r}^K$ for all $K$ in $\mathcal{T}_h$
5   $\delta\hat{\mathbf{u}} \leftarrow$ Solve $\mathbf{H}\delta\hat{\mathbf{u}} = \mathbf{r}$
6   begin Obtain $\delta\mathbf{q}$ and $\delta\mathbf{u}$ from $\delta\hat{\mathbf{u}}$
7       for $K$ in $\mathcal{T}_h$ do
8           $\delta\mathbf{q}^K$, $\delta\mathbf{u}^K \leftarrow$ Recover element solution from $\delta\hat{\mathbf{u}}^{\partial K}$
9       $\delta\mathbf{q}$, $\delta\mathbf{u} \leftarrow$ Assemble $\delta\mathbf{q}^K$ and $\delta\mathbf{u}^K$ for all $K$ in $\mathcal{T}_h$
```

III.C. Proposed sequential implementation

The sequential implementation of the HDG solver can be summarized by Algorithm 6. Specifically, from $\mathbf{q}$, $\mathbf{u}$, and $\hat{\mathbf{u}}$ we want to obtain $\delta\mathbf{q}$, $\delta\mathbf{u}$, and $\delta\hat{\mathbf{u}}$. To this end, matrix $\mathbf{H}$ and vector $\mathbf{r}$ are computed element-by-element, Lines 1-4. Then, the global linear system is solved in Line 5. Finally, the obtained value of $\delta\hat{\mathbf{u}}$ allows obtaining $\delta\mathbf{q}$ and $\delta\mathbf{u}$ in an element-wise manner, Lines 6-9.

It is important to point out that all the previous element-by-element loops can be performed in parallel, since the computations on each element do not depend on the rest of elements. Specifically, the computation of the element: quantities, inverses in Equation (12), contributions in Equations (13) and (14), and the local linear systems in Equation (15). In addition, the global linear system can be solved with standard distributed and parallel method. Taking into account these observations, we present in Section IV the proposed distributed and parallel implementation of the HDG solver.

IV. HDG method: distributed and parallel implementation

To solve the non-linear weak formulation, we use Newton's method with a backtracking line search. Specifically, we consider the following Newton iteration. First, we obtain a partition of the mesh that takes into account the stencil of the HDG method, see Section IV.A. This partition is used to determine the distribution maps required to parallelize the HDG method. Second, we compute in parallel the elemental quantities required to solve the linear system in Equation (7). Third, we obtain the solution increments $\delta\hat{\mathbf{u}}, \delta\mathbf{q}$, and $\delta\mathbf{u}$ by solving the linear system in Equation (7). According to III.B, we solve the linear system in two stages. In the first stage we obtain $\delta\hat{\mathbf{u}}$ by solving in parallel the global linear system in Equation (9), see Section IV.C. In the second stage, we obtain $\delta\mathbf{q}$ and $\delta\mathbf{u}$ by evaluating in parallel the element-wise expression in Equation (15), see Section IV.C.2. Then, we obtain a scaling scalar α by means of a bactracking line search method. The scaling value α ensures a sufficient decrease of the norm of the residual $\mathbf{r}$ and therefore, it helps to improve the global convergence of the method. Finally, α and $\delta\hat{\mathbf{u}}, \delta\mathbf{q}$, and $\delta\mathbf{u}$ are used to update the solution vectors $\mathbf{q} \leftarrow \mathbf{q} + \alpha\delta\mathbf{q}, \mathbf{u} \leftarrow \mathbf{u} + \alpha\delta\mathbf{u}$, and $\hat{\mathbf{u}} \leftarrow \hat{\mathbf{u}} + \alpha\delta\hat{\mathbf{u}}$.

IV.A. Partition of the HDG mesh

The goal is to form and solve in parallel the linear system arising from the HDG method. To this end, we need to partition the degrees of freedom in a load balanced manner. Note that in the HDG method a degree of freedom is determined by the discontinuous approximation spaces on the elements and faces, see Section II.A.3. That is, a degree of freedom is either on an element or a face. If we group the degrees of freedom by elements and faces, we need only a partition in colors of the elements and faces of the mesh. To this end, we assign a global numbering to all the elements and faces of the mesh. Then, we express each element of the mesh in terms of the global numbering of its bounding faces. With the resulting connectivity graph, we use METIS[22] to obtain a mesh partition that groups elements and faces of the same color. Then, the color of a degree of freedom is determined by the color of either the element or the face that owns it. This mesh partition is used later to compute in parallel the elemental quantities required to form the Jacobian and residual. In addition, this partition is used to solve the two stages of the HDG linear system. First, the

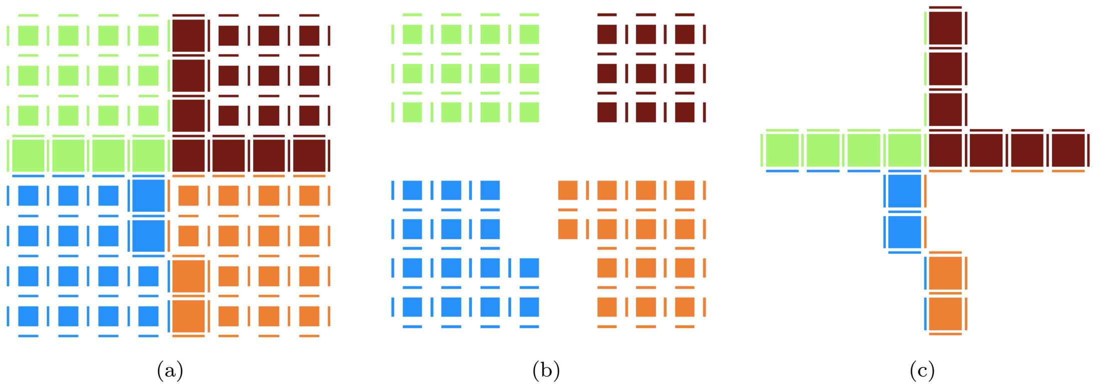


Figure 2. Partition of the elements and faces of a HDG mesh in four colors: (a) full partition, (b) interior by colors, and (c) interface by colors.

partition is used to assign colors to the matrix and vector rows. Specifically, the rows in color a correspond to the degrees of freedom of the group of elements (faces) that also have color a. This color map allows the distribution of the rows in several processors. In this way, the operations with sparse matrices and vectors, required by the pre-conditioned iterative solver, can be performed in parallel, see Section IV.C.1. Second, this partition is used to solve the element level linear systems in parallel for all the colors, see Section IV.C.2.

The communication requirements of the parallel implementation are determined by the structure of the connections between the degrees of freedom of the HDG method. Specifically, two degrees of freedom are connected (*adjacent*) if they both appear with a non-zero coefficient in at least one equation. Since all the degrees of freedom are either on an element or on a face, the connection of structure can be expressed in terms of elements and faces. That is, all the possible adjacencies of the HDG method are described by the following cases: an element K is *adjacent* to all the faces that are on ∂K; a face is *adjacent* to an element K if it is on ∂K; and two faces are *adjacent* if they are adjacent to the same element.

To describe the parallel implementation in terms of the HDG adjacencies and the mesh partition, we consider the following definitions:

- ${\mathcal{T}_h}^a, {\mathcal{E}_h}^a$: on-processor elements and faces of color a.
- ${\mathcal{T}_h}^{a,a}, {\mathcal{E}_h}^{a,a}$: on-processor elements and faces of color a only adjacent to faces of color a (interior).
- ${\mathcal{T}_h}^{a,b}, {\mathcal{E}_h}^{a,b}$: on-processor elements and faces of color a that have an adjacent face of color b (interface with processor b).
- ${\mathcal{T}_h}^{a,*}, {\mathcal{E}_h}^{a,*}$: on-processor elements and faces of color a that have an adjacent face with different color (interface of processor a with other processors).
- ${\mathcal{T}_h}^{*,a}, {\mathcal{E}_h}^{*,a}$: off-processor elements and faces that have an adjacent face with of color a (interface of other processors with processor a).
- Color a is *adjacent* to color b if ${\mathcal{T}_h}^{a,b}$ or ${\mathcal{E}_h}^{a,b}$ are not empty.

IV.B. Parallel implementation

The parallel implementation of the HDG method that corresponds to Algorithm 6 is presented in Algorithm 8. Before describing the algorithm, we want to point out that underlined vectors denote that their components are distributed among the memory that corresponds to each processor. That is, $\underline{\mathbf{q}}$, $\underline{\mathbf{u}}$, $\underline{\hat{\mathbf{u}}}$, $\underline{\delta\mathbf{q}}$, $\underline{\delta\mathbf{u}}$, $\underline{\delta\hat{\mathbf{u}}}$ are vectors with distributed components. The parallel distributed implementation of the HDG method seeks to obtain $\underline{\delta\mathbf{q}}$, $\underline{\delta\mathbf{u}}$, $\underline{\delta\hat{\mathbf{u}}}$ from $\underline{\mathbf{q}}$, $\underline{\mathbf{u}}$, $\underline{\hat{\mathbf{u}}}$. To this end, each processor executes Line 1 to obtain the current processor number. Then, the distributed matrix $\underline{\mathbf{H}}$ and vector $\underline{\mathbf{r}}$ are computed from $\underline{\mathbf{q}}$, $\underline{\mathbf{u}}$, $\underline{\hat{\mathbf{u}}}$, Lines 2-6. Then, the global linear system is solved with a distributed solver, Line 7. This operation is performed by a GMRES solver pre-conditioned with an ASDD(l) and an ILU(k) factorization as an approximated sub-domain problem. Finally, $\underline{\delta\mathbf{q}}$ and $\underline{\delta\mathbf{u}}$ are obtained from $\underline{\delta\hat{\mathbf{u}}}$, Lines 8-12.

Algorithm 2: Distributed solver for the HDG linear system

```
   Input: $\underline{\mathbf{q}}$, $\underline{\mathbf{u}}$, $\underline{\hat{\mathbf{u}}}$
   Output: $\underline{\delta\mathbf{q}}$, $\underline{\delta\mathbf{u}}$, $\underline{\delta\hat{\mathbf{u}}}$
1  $a \leftarrow$ Current processor
2  begin Compute distributed matrix $\underline{\mathbf{H}}$ and vector $\underline{\mathbf{r}}$ from $\underline{\mathbf{q}}$, $\underline{\mathbf{u}}$, $\underline{\hat{\mathbf{u}}}$
3      Gather $\mathbf{q}^K$, $\mathbf{u}^K$, $\hat{\mathbf{u}}^{\partial K}$ from $\underline{\mathbf{q}}$,$\underline{\mathbf{u}}$,$\underline{\hat{\mathbf{u}}}$ for all $K$ in ${\mathcal{T}_h}^a$
4      for $K$ in ${\mathcal{T}_h}^a$ do
5          $\mathbf{H}^K$, $\mathbf{r}^K \leftarrow$ Elemental contributions from $\mathbf{q}^K$, $\mathbf{u}^K$, $\hat{\mathbf{u}}^{\partial K}$
6      $\underline{\mathbf{H}}$, $\underline{\mathbf{r}} \leftarrow$ Assemble $\mathbf{H}^K$ and $\mathbf{r}^K$ for all $K$ in ${\mathcal{T}_h}^a$
7  $\underline{\delta\hat{\mathbf{u}}} \leftarrow$ Distributed solve $\underline{\mathbf{H}}\underline{\delta\hat{\mathbf{u}}} = \underline{\mathbf{r}}$
8  begin Obtain $\underline{\delta\mathbf{q}}$ and $\underline{\delta\mathbf{u}}$ from $\underline{\delta\hat{\mathbf{u}}}$
9      Gather $\delta\hat{\mathbf{u}}^{\partial K}$ from $\underline{\delta\hat{\mathbf{u}}}$ for all $K$ in ${\mathcal{T}_h}^a$
10     for $K$ in ${\mathcal{T}_h}^a$ do
11         $\delta\mathbf{q}^K$, $\delta\mathbf{u}^K \leftarrow$ Recover element solution from $\delta\hat{\mathbf{u}}^{\partial K}$
12     $\underline{\delta\mathbf{q}}$, $\underline{\delta\mathbf{u}} \leftarrow$ Distribute $\delta\mathbf{q}^K$ and $\delta\mathbf{u}^K$ for all $K$ in ${\mathcal{T}_h}^a$
```

It is important to highlight that there are two main differences with the sequential solver. First, the element-wise loops are scatter among processors. Second, the components of the vectors have to be scattered and gathered, Lines 3 and 9. Basically, there are two types of vectors according to the type of entity: the DOFs on the elements; and the DOFs on the faces. First, the element vectors are distributed in sub-domains of elements. Moreover, it is required to: gather $\mathbf{q}^K$, $\mathbf{u}^K$ from $\underline{\mathbf{q}}$ and $\underline{\mathbf{u}}$ for all K in ${\mathcal{T}_h}^p$; and distribute $\delta\mathbf{q}^K$ and $\delta\mathbf{u}^K$ for all K in ${\mathcal{T}_h}^p$ to form $\underline{\delta\mathbf{q}}$, and $\underline{\delta\mathbf{u}}$, respectively. To this end, each processor p stores the components associated with the elements in ${\mathcal{T}_h}^p$ (element sub-domains). Second, the face vectors are distributed in sub-domains of faces. Furthermore, it is required to gather $\hat{\mathbf{u}}^{\partial K}$ from $\underline{\hat{\mathbf{u}}}$ and $\delta\hat{\mathbf{u}}^{\partial K}$ for all K in ${\mathcal{T}_h}^p$. To this end, each processor p stores the components associated with the faces in ${\mathcal{E}_h}^p$ (face sub-domains). Specifically, a processor a has to receive from a processor b the components associated with the faces in ${\mathcal{E}_h}^b$ that are also in $\partial{\mathcal{T}_h}^a$.

IV.C. Solve the linear system

As we have seen in Section III.A, the linear system that arise from HDG can be solved in two stages:

IV.C.1. Solving the global linear system

The first stage corresponds to solve the linear system with unknowns $\delta\hat{\mathbf{u}}$ on the faces, Equation (9). To solve this linear system, we use the Trilinos[23] library. Specifically, we use the provided Generalized Minimal RESidual (GMRES) iterative solver pre-conditioned with an algebraic additive Schwarz domain decomposition with l-levels of overlap. Moreover, each one of the overlapped sub-problems is approximated with an ILU(k) factorization. Note that the domains of the decomposition are determined by the mesh partition used to distribute the problem.

The GMRES solver is an iterative solver of the family of the Krylov methods. Recall that Krylov methods are projection (Galerkin) techniques for solving linear systems

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

and they are based on the generation of the Krylov subspace

$$\mathcal{K}_j := \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{j-1}\mathbf{r}_0\},$$

where $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}$. The main advantage of Krylov methods is that they allow solving large linear systems, since the have lower memory footprint than direct solvers. Specifically, the GMRES method with restart helps to reduce even more the memory footprint.

The cost of GMRES is dominated by the computation of a sparse matrix-vector product at each iteration. It is important to point out that the sparse matrix-vector product can be parallelized to improve the performance of the iterative solver. Furthermore, GMRES can be pre-conditioned to reduce the number of iterations. To this end, an algebraic additive Scharz domain decomposition with l-levels of overlap (ASDD(l)) is used. In addition, the ASDD(l) is combined with an approximate solver for the sub-domains. Specifically, with an incomplete LU factorization with k-levels of fill-in (ILU(k)).

Note that ASDD(l) is a divide-and-conquer approach to pre-condition a linear system. Therefore, it is suitable for distributed parallel computations. To this end, it is required to obtain a mesh partition that determines n_p sub-domains from the graph of DOFs connections. The ASDD(l) pre-conditioners is the operator:

$$\mathbf{M}^{-1} = \sum_{p=1}^{n_p} \mathbf{R}_p^T \tilde{\mathbf{A}}_p^{-1} \mathbf{R}_p,$$

where n_p is the number of sub-domains (processors), $\mathbf{R}_p$ restricts a vector to the p-th sub-domain (Boolean matrix), and $\tilde{\mathbf{A}}_p$ approximates $\mathbf{R}_p \mathbf{A} \mathbf{R}_p^T$.

In this work, $\tilde{\mathbf{A}}_p$ is an incomplete LU factorization with k-levels of fill-in (ILU(k)). Specifically, an LU factorization where the entries out of the original sparsity pattern of $\tilde{\mathbf{A}}_p$ are drop. Furthermore, in the case of k-levels of fill-in the sparsity pattern is enhanced to ensure that the neighbours at distance k of each degree of freedom are also considered.

Finally, to reduce the number of iterations of the domain decomposition pre-conditioner, l-levels of sub-domain *overlapping* are considered. That is, each sub-domain is augmented with the DOFs on the faces of other sub-domains that are at distance l (trhough DOF connections) of the current sub-domain. Note that zero-levels corresponds to a block Jacobi pre-conditioner where each block is determined by the DOFs on the faces of each sub-domain.

IV.C.2. Solving the local linear system

Once the solution vector $\delta\hat{\mathbf{u}}$ is obtained with the desired accuracy, we obtain the element-wise solution vectors $\delta\mathbf{q}$ and $\delta\mathbf{u}$ by evaluating in parallel for all processors the element-wise expression in Equation (8). It is straigtforward to see that the sequential version of this stage scales linearly with the number of elements. Furthermore, it can be element-wise parallelized. Therefore, the cost of solving the HDG linear system, for meshes composed by a large number of elements, is dominated by the first stage.

V. Results: SD7003 airfoil, $\alpha = 5.0^\circ$, $Re = 10000$, $M_\infty = 0.1$

In the following examples, the weak scaling of the considered global linear system solver and an unsteady flow that presents a periodic vortex shedding regime are studied. To this end, we consider the compressible Navier-Stokes solution for the flow around an SD7003 airfoil at an angle of attack of 5.0°, with a Reynolds number $Re = 10000$ and a free-stream Mach number of $M_\infty = 0.1$. All the results have been obtained on a computer with four eight-core AMD Opteron 6320 CPUs, each one with a clock frequency of 2.8GHz, and 24MB of cache, and a total memory of 512 GBytes. The solver has been fully developed with the interpreted language Python equipped with the SciPy, Numpy, and PyTrilinos libraries. Before performing the simulations, several C-type structured meshes composed by high-order triangular elements have been generated. The code corresponds to the implementation presented in this work, and uses a diagonally implicit Runge-Kutta (DIRK) of order 3 in 3 stages for time stepping. The Newton solver use a convergence tolerance of 10^{-8} and a backtracking line search method. In each Newton iteration, a GMRES solver with convergence tolerance of 10^{-12} that restarts every 30 iterations is called. The iterative solver uses an algebraic additive Schwarz domain decomposition with or without overlap, and where the sub-domain solver corresponds to an incomplete LU factorization with or without fill-in.

Several parameters have been varied to obtain the results. First, the code has been launched with different number of processors (n_p), that also correspond to the number of sub-domains. To this end, meshes with different interpolation degree (p) and number of elements (n_e) and faces (n_f) have been generated. Depending on the test case, different time steps (dt) for the numerical time integration have been used. Finally, four pre-conditioning options have been considered. Specifically, the additive Schwarz domain decomposition has been run with l-levels of overlap, for $l = 0, 1$. Furthermore, the sub-domain ILU has been tested with k-levels of fill-in, for $k = 0, 1$.

				Number of iterations of one linear solve			
				ASDD(0)		ASDD(1)	
n_p	n_e	n_f	dt	ILU(0)	ILU(1)	ILU(0)	ILU(1)
1	192	308	0.0056	28	18	28	18
2	408	641	0.0040	40	31	26	19
4	768	1192	0.0028	60	59	39	26
8	1540	2368	0.0020	111	112	29	23
16	3072	4688	0.0014	135	127	35	26
32	6164	9358	0.0010	109	106	39	26

Table 1. Weak scaling according to the number of processors $n_p = 1, 2, 4, 8, 16, 32$ (≈ 192 elements and ≈ 288 faces per processor): number of iterations of the GMRES linear solver with an ASDD(l)/ILU(k) pre-conditioner.

				Total time of one linear solve (s.)			
				ASDD(0)		ASDD(1)	
n_p	n_e	n_f	dt	ILU(0)	ILU(1)	ILU(0)	ILU(1)
1	192	308	0.0056	0.52	5.59	0.69	5.77
2	408	641	0.0040	0.60	5.98	0.70	6.40
4	768	1192	0.0028	0.63	5.73	0.74	6.19
8	1540	2368	0.0020	0.90	6.00	0.75	6.46
16	3072	4688	0.0014	1.23	6.32	0.95	6.68
32	6164	9358	0.0010	2.03	8.20	1.62	8.92

Table 2. Weak scaling according to the number of processors $n_p = 1, 2, 4, 8, 16, 32$ (≈ 192 elements and ≈ 288 faces per processor): total time of the GMRES linear solver with an ASDD(l)/ILU(k) pre-conditioner.

V.A. Scaling of the global linear system

The HDG linear system requires at each Newton iteration the solution of a global linear system that corresponds to the DOFs on the mesh faces. In this test case, the weak scaling according to the number of processors of the corresponding linear solver is considered. Specifically, how the number of iterations and the total time for the linear solve scales when the problem size is doubled.

To obtain the results for this test case, four possible pre-conditioners have been used with a different number of processors, $n_p = 1, 2, 4, 8, 16, 32$. The different pre-conditioners are characterized by varying the number of levels of overlap and fill-in, $l, k = 0, 1$. That is, additive Schwarz domain decomposition (ASDD) with (ASDD(1)) or without (ASDD(0)) overlap, and ILU(0) or ILU(1) have been combined. Note that for one processor ($n_p = 1$), ASDD(0) and ASDD(1) are the same pre-conditioner since there is not an interface between DOFs on different processors. According to the number of processors, six different meshes of interpolation degree $p = 3$ have been generated. Each mesh is composed by a number of elements that ensures that, after mesh partitioning, each sub-domain is composed by approximately 192 elements and 288 faces. To ensure that the comparison is fair, the time step is divided by $\sqrt{2}$ each time the number of mesh elements is doubled. That is, the time step is related with the average element size to ensure a system of equivalent difficulty for the different meshes.

The weak scaling results for the number of iterations of one linear solve is presented in Table 1. The results show that if ASDD(0) is used, the number of iterations grows when the number of processors and elements is doubled. On the contrary, the number of iterations remains more stable when ASDD(1) is used. Finally, is important to point out that ILU(1) always help to reduce the number of iterations but, less significantly than ASDD(1).

To decide between one of the four explored combinations of pre-conditioner, it is also important to consider the total time of the linear solver. To this end, Table 2 presents the results for the weak scaling of the total time for the four possible pre-conditioners. The results show that pre-conditioners that use a sub-domain solver with ILU(1) are more expensive that the corresponding ILU(0) combinations. The main reason, is that the resulting sub-domain approximate solver requires a matrix with more non-zero entries and therefore, the creation of the pre-conditioner and its application to a vector (backward and forward substitution) are more expensive. On the contrary, the use of 1-level of overlap on the ASDD increases the time but less significantly than the use of 1-level of fill-in for the ILU.

It is also important to point out that this code has been executed in a computer node with 32 cores and 512 GBytes of shared-memory. Therefore, the solver time degenerates when more processors are used. This is due to the fact the memory BUS gets more charge when more processors are used. Specifically, the memory bandwidth of the computer is under-used for 1 and 2 cores, efficiently used for 4 or 8 cores, and over-used for 16 or 32 cores.

To choose the option that scales better, one has to take into account the weak scaling for the number of iterations and the total time of one linear solve. On the one hand, the ASDD with one-level of overlap shows a stable number of iterations. On the other hand, the ILU(0) requires smaller computational time than ILU(1). Therefore, we can conclude that ASDD(1)/ILU(0) is the option that scales better for the considered application and computer.

V.B. Vortex shedding regime

In this test case, a compressible Navier-Stokes solution of the flow around a SD7003 airfoil at an angle of attack of $5.0°$, with a Reynolds number $Re = 10000$ and a free-stream Mach number of $M_\infty = 0.1$, is presented. In this regime, the flow involves the formation of a laminar separation bubble along its upper surface that results in periodic vortex shedding. This regime has already been studied with a high-order discontinuous Galerkin solver by Uranga *et al.* in Ref. [24,25]. Herein, we have run the proposed high-order and parallel HDG solver with 16 cores. Accordingly, the mesh has been partitioned in 16 sub-domains. Specifically, the mesh is composed by 1728 elements and 2656 faces of interpolation degree $p = 5$. This results in 311040 DOFS on the elements and 63744 DOFs on the faces. Then, the time integration has been performed using a time step $dt = 0.05$. Finally, the global linear system has been pre-conditioned with ASDD(1)/ILU(0).

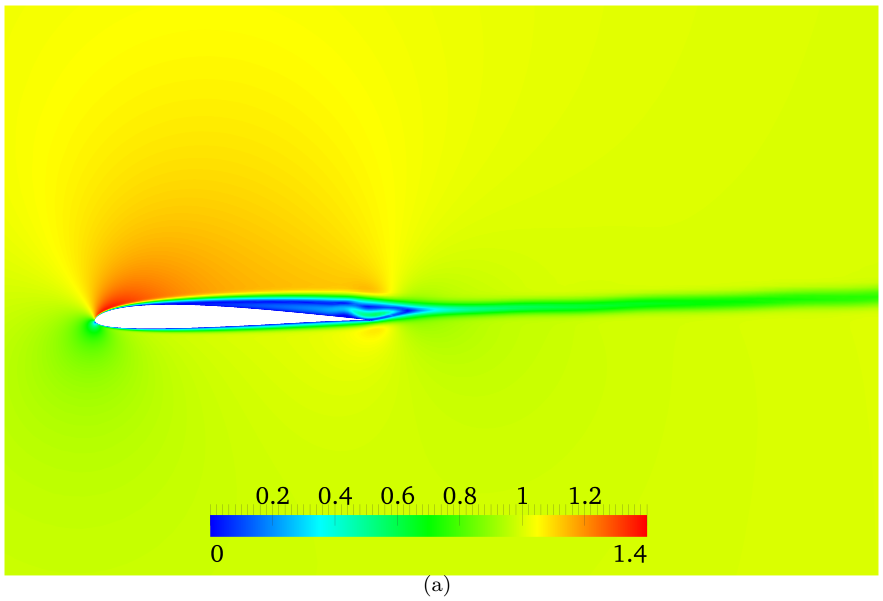
The velocity magnitude for this flow case is presented at two different times in Figures 3(a) and 3(b). On the one hand, Figure 3(a) shows that the flow around the airfoil at time $t = 5.0$ is laminar and the separation bubble is still growing. On the other hand, Figure 3(b) shows the periodic vortex shedding regime at time $t = 7.5$. Note that the first vortex is released approximately at time $t = 5.5$.

VI. Concluding remarks

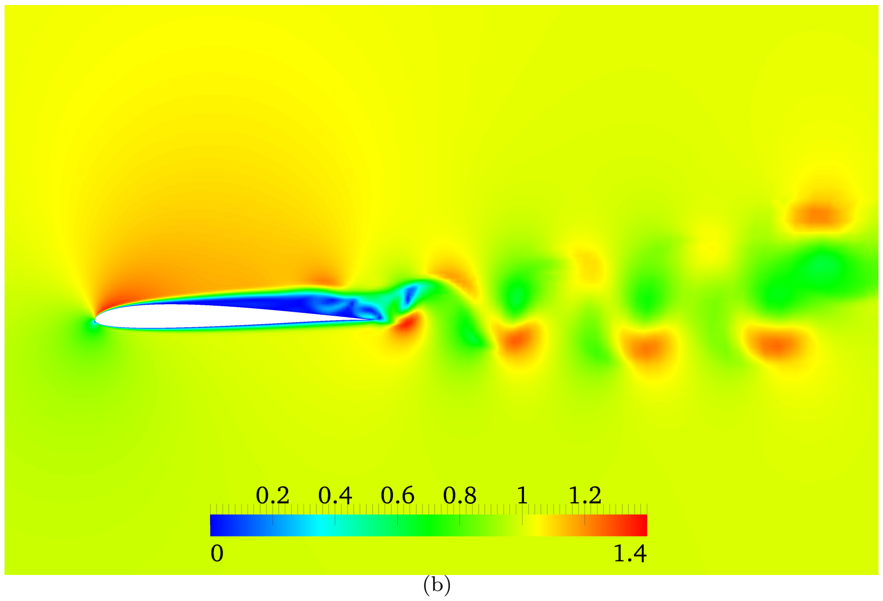
We have seen that HDG method leads to linear systems that have a structure that can be exploited in terms of a parallel implementation. The resulting linear systems can be solved in two stages. A first stage where a sparse linear system composed by dense blocks is obtained. This linear system has to be solved with a standard distributed and parallel linear solver. However, this system has the main advantage that it only depends on the degrees of freedom on the boundaries of the elements. This reduction of the global degrees of freedom results in a smaller computational cost. The second stage allows obtaining the values of the conserved quantities and their gradients in terms of the solution on the element boundaries. This operation can be performed independently for all the elements of the mesh. Therefore, it scales linearly with the number of elements and it can be parallelized. We exploit this characteristic to solve the second stage of the linear system in parallel and with small communication. Therefore, the cost of each Newton iteration is dominated by the cost of solving the global linear system. To reduce the cost of of solving the global linear system, we have considered a distributed solver that consists on GMRES with restart pre-conditioned with an algebraic additive Schwarz domain decomposition with l-levels of overlap (ASDD(l)). To approximate the sub-domains problems, we have used an incomplete LU factorization with k-levels of fill-in (ILU(k)). Taking into account the number of iterations and the total time per linear solve, we have concluded that GMRES with restart pre-conditioned with ASDD(1)/ILU(0) presents the best weak scaling. To show the possibilities of the proposed parallel implementation of the HDG method, we have included the study of the unsteady flow around an airfoil that presents periodic vortex shedding.

Acknowledgments

X. Roca, N. C. Nguyen, and J. Peraire would like to acknowledge the Singapore-MIT Alliance for the MOU contract and the Air Force Office of Scientific Research under the grant FA9550-12-0357.



(a)



(b)

Figure 3. Velocity magnitude for the SD7003 airfoil ($\alpha = 5.0^\circ$, $Re = 10000$, and $M_\infty = 0.1$) at: (a) $t = 5.0$; and (b) $t = 7.5$.

References

[1]Bassi, F. and Rebay, S., "A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations," *J. Comput. Phys.*, Vol. 131, No. 2, 1997, pp. 267–279.

[2]Peraire, J. and Persson, P., "The compact discontinuous Galerkin (CDG) method for elliptic problems," *SIAM J. Sci. Comput.*, Vol. 30, No. 4, 2008, pp. 1806–1824.

[3]Cockburn, B., Gopalakrishnan, J., and Lazarov, R., "Unified hybridization of discontinuous Galerkin, mixed and continuous Galerkin methods for second order elliptic problems," *SIAM J. Numer. Anal.*, Vol. 47, 2009, pp. 1319–1365.

[4]Cockburn, B., Dong, B., and Guzmán, J., "A superconvergent LDG-hybridizable Galerkin method for A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems," *Math. Comp.*, Vol. 77, 2008, pp. 1887–1916.

[5]Cockburn, B., Dong, B., Guzmán, J., Restelli, M., and Sacco, R., "A Hybridizable Discontinuous Galerkin Method for Steady-State Convection-Diffusion-Reaction Problems," *SIAM J. Sci. Comput.*, Vol. 31, 2009, pp. 3827.

[6]Nguyen, N., Peraire, J., and Cockburn, B., "An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations," *J. Comput. Phys.*, Vol. 228, 2009, pp. 3232–3254.

[7]Nguyen, N., Peraire, J., and Cockburn, B., "An implicit high-order hybridizable discontinuous Galerkin method for nonlinear convection-diffusion equations," *J. Comput. Phys.*, Vol. 228, 2009, pp. 8841–8855.

[8]Cockburn, B., Gopalakrishnan, J., Nguyen, N., Peraire, J., and Sayas, F.-J., "Analysis of an HDG method for Stokes flow," *Math. Comp.*, To appear.

[9]Nguyen, N., Peraire, J., and Cockburn, B., "A hybridizable discontinuous Galerkin method for the incompressible Navier-Stokes equations," *Proceedings of the 48th AIAA Aerospace Sciences Meeting and Exhibit*, 2010, pp. AIAA Paper 2010–362.

[10]Nguyen, N., Peraire, J., and Cockburn, B., "Hybridizable discontinuous Galerkin methods," *Spectral and High Order Methods for Partial Differential Equations*, edited by J. S. Hesthaven and E. M. Ronquist, Vol. 76, 2011, pp. 63–84.

[11]Nguyen, N., Peraire, J., and Cockburn, B., "An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier-Stokes equations," *J. Comput. Phys.*, In press.

[12]Nguyen, N., Peraire, J., and Cockburn, B., "A comparison of HDG methods for Stokes flow," *Journal of Scientific Computing*, Vol. 45, 20010, pp. 215–237.

[13]Peraire, J., Nguyen, N., and Cockburn, B., "A hybridizable discontinuous Galerkin method for the compressible Euler and Navier-Stokes equations," *Proceedings of the 48th AIAA Aerospace Sciences Meeting and Exhibit*, 2010, pp. AIAA Paper 2010–363.

[14]Smith, B., Bjorstad, P., and Gropp, W., *Domain decomposition*, Cambridge University Press, 2004.

[15]Persson, P. and Peraire, J., "Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations," *SIAM J. Sci. Comput.*, Vol. 30, No. 6, 2008, pp. 2709–2733.

[16]Cockburn, B. and Shu, C.-W., "Runge-Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems," *J. Sci. Comput.*, Vol. 16, No. 3, 2001, pp. 173–261.

[17]Fidkowski, K., Oliver, T., Lu, J., and Darmofal, D., "p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations," *J. Comput. Phys.*, Vol. 207, 2005, pp. 92–113.

[18]Diosady, L. T. and Darmofal, D. L., "Preconditioning methods for discontinuous Galerkin solutions of the Navier–Stokes equations," *Journal of Computational Physics*, Vol. 228, No. 11, 2009, pp. 3917–3935.

[19]Diosady, L., *Domain Decomposition Preconditioners for Higher-Order Discontinuous Galerkin Discretizations*, Ph.D. thesis, M.I.T., September 2011.

[20]Roca, X., Nguyen, N., and Peraire, J., "GPU-accelerated sparse matrix-vector product for a hybridizable discontinuous Galerkin method," *49th AIAA Aerospace Sciences Meeting and Exhibit*, 2011.

[21]Anderson, D., Tannehill, J., and Pletcher, R., *Computational fluid mechanics and heat transfer*, Hemisphere Publishing, New York, NY, 1984.

[22]Karypis, G. and Kumar, V., *Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0*, 1995.

[23]Heroux, M. A. and Willenbring, J. M., "Trilinos Users Guide," Tech. Rep. SAND2003-2952, Sandia National Laboratories, 2003.

[24]Uranga, A., Persson, P.-O., Drela, M., and Peraire, J., "Implicit Large Eddy Simulation of transitional flows over airfoils and wings," *Proceedings of the 19th AIAA Computational Fluid Dynamics*, Vol. 4131, 2009.

[25]Uranga, A., Persson, P.-O., Drela, M., and Peraire, J., "Implicit Large Eddy Simulation of transition to turbulence at low Reynolds numbers using a Discontinuous Galerkin method," *International Journal of Numerical Methods in Engineering*, Vol. 86, 2011.