Matlab tools for HDG in three dimensions

Zhixing Fu,* Luis F. Gatica,† Francisco–Javier Sayas‡

May 7, 2013

Abstract

In this paper we provide some Matlab tools for efficient vectorized coding of the Hybridizable Discontinuous Galerkin for linear variable coefficient reaction-diffusion problems in polyhedral domains. The resulting tools are modular and include enhanced structures to deal with convection-diffusion problems, plus several projections and a superconvergent postprocess of the solution. Loops over the elements are exclusively local and, as such, have been parallelized.

1   Introduction

In this paper we provide some programming tools for full Matlab implementation of the Hybridizable Discontinuous Galerkin (HDG) method on general conforming tetrahedral meshes for fixed but arbitrary polynomial degree. The presentation is detailed on a second order linear reaction-diffusion equation with variable coefficients and mixed boundary conditions, but we also provide the tools to construct the matrices needed for convection-diffusion problems with variable convection, thus creating all necessary blocks to deal with general steady-state problems.

The HDG method originated in a sequence of papers of Bernardo Cockburn and his collaborators, consolidating in the unified framework of [4]. As seen in that paper, the HDG can be considered as a Mixed Finite Element Method (MixedFEM) [2], coded with the use of Lagrange multipliers to weakly enforce the restrictions on interelement faces [1], and then hybridized so that the only global variable is the collection of Lagrange multipliers, that ends up being an optimal approximation of the primal variable on the faces of the triangulation.

Compared with general MixedFEM (programmed in hybridized form), HDG has the advantage of not using degrees of freedom to stabilize the discrete equations, while keeping equal optimal order of convergence in all computed fields. (Stability is obtained through

a stabilization parameter.) From this point of view, HDG is a valid option if one is willing to pay the prize of using MixedFEM, for instance, to obtain approximations of more fields of the solution of the problem. In comparison with other FEM, that work directly on the second order formulation, HDG performs well for high orders [9]. For low order methods, HDG can be adopted in situations where either MixedFEM or DG are thought to be advantageous. We will, however, not exploit here the advantages of HDG/DG for having non-conforming meshes or variable degree. Extension of the code to variable-degree methods does not seem to change much, but it requires rethinking the data structures and the vectorization process. Extension to more general meshes would require new tools for the geometric handling that we are not dealing with at this moment. In comparison to other DG methods (mainly those of the Interior Penalty family), HDG requires less degrees of freedom in the solution of the global system, since this has been reduced to the interfaces of the elements. On the positive side as well, HDG does not contain any penalization parameter that needs tuning to obtain convergence. It also has some attractive superconvergence properties that allow for local element-by-element postprocessing *à la Stenberg* [10].

One goal of the paper is the systematization of the construction of local and global matrices by looping over quadrature nodes and polynomial degrees, avoiding large loops over elements. We partially accomplish this by using Matlab inbuilt functions for Kronecker products, construction of sparse matrices, and vectorization. All loops on elements are purely local and have been parallelized so that they can take advantage of the Matlab Parallel Toolbox. We hope this piece of work will contribute to the popularization of a method that has already a sizeable follow-up, given its good properties. This being Matlab code, we are not expecting the code to run on very large problems, but, as we show in the experiments, we can show reasonably high order of convergence for three dimensional problems, working on a laptop with only two processors. We will also comment on how this code can be easily modified to provide the hybridized implementation of the Brezzi–DOuglas–Marini (BDM) mixed element, which is an alternative to using $\mathbf{H}(\mathrm{div})$-conforming bases [6].

Model equations. Let $\Omega \subset \mathbb{R}^3$ be a polyhedron with boundary Γ, divided into a Dirichlet and a Neumann part (Γ_D and Γ_N) such that each part is the union of faces of Γ. The unit outward-poiting normal vector field on Γ is denoted $\boldsymbol{\nu}$. The model problem we will be discussing in this document is

$$\kappa^{-1}\boldsymbol{q} + \nabla u = 0 \qquad \text{in } \Omega, \tag{1.1a}$$

$$\nabla \cdot \boldsymbol{q} + c\,u = f \qquad \text{in } \Omega, \tag{1.1b}$$

$$u = u_D \qquad \text{on } \Gamma_D, \tag{1.1c}$$

$$-\boldsymbol{q} \cdot \boldsymbol{\nu} = \boldsymbol{g}_N \cdot \boldsymbol{\nu} \qquad \text{on } \Gamma_N. \tag{1.1d}$$

The diffusion coefficient κ is strictly positive, while $c \geqslant 0$. (Both of them are functions of the space variables.) The Neumann boundary condition is given in non-standard way, as the normal component of a vector field (of which only the normal component is used), in order to give an easier way to test exact solutions. Modifications for the case of a scalar field are straightforward.

Discrete elements. The Hybridizable Discontinuous Galerkin method that we will describe here is based on regular tetrahedrizations of the domain. We thus consider a tetrahedral partition of Ω, $\mathcal{T}_h$. The set of all faces in the triangulation is denoted $\mathcal{E}_h$, with the subsets $\mathcal{E}_h^{\text{int}}$, $\mathcal{E}_h^D$, $\mathcal{E}_h^N$ corresponding to interior, Dirichlet and Neumann faces. For convenience, we will write

$$N_{\text{elt}} = \#\mathcal{T}_h, \qquad N_{\text{fc}} = \#\mathcal{E}_h, \qquad N_{\text{dir}} = \#\mathcal{E}_h^D, \qquad N_{\text{neu}} = \#\mathcal{E}_h^N.$$

Upon numbering, we can identify elements and faces with respective index sets

$$\mathcal{T}_h \equiv \{1, \ldots, N_{\text{elt}}\}, \qquad \mathcal{E}_h \equiv \{1, \ldots, N_{\text{fc}}\}.$$

This will allow us to write some computational expressions in a format that is very close to their mathematical definition. The local spaces for discretization of u and $\boldsymbol{q}$ are those of trivariate polynomials of degree up to k. The global description of these spaces is

$$W_h := \prod_{K \in \mathcal{T}_h} \mathcal{P}_k(K), \qquad \boldsymbol{V}_h := W_h^3 = \prod_{K \in \mathcal{T}_h} \mathcal{P}_k(K)^3.$$

There is a third space, defined on the skeleton of the tetrahedrization:

$$M_h := \prod_{e \in \mathcal{E}_h} \mathcal{P}_k(e),$$

where $\mathcal{P}_k(e)$ is the space of bivariate polynomials of degree not larger than k on tangential coordinates. Integral notation will always be given as

$$(u, v)_K := \int_K u\,v, \qquad (\boldsymbol{q}, \boldsymbol{r})_K := \int_K \boldsymbol{q} \cdot \boldsymbol{r}, \qquad \langle u, v \rangle_{\partial K} := \int_{\partial K} u\,v.$$

Similarly, we will use terms like $\langle u, v \rangle_e$ with $e \in \mathcal{E}_h$, $\langle u, v \rangle_{\Gamma_D}$, and $\langle u, v \rangle_{\Gamma_N}$. On the boundary of a given element, the normal vector $\boldsymbol{\nu}_K$ will point outwards. However, when it is clear what the element is, we will simply write $\boldsymbol{\nu}$.

HDG. A key ingredient of HDG is a stabilization function. This is a non-negative piecewise constant function on the boundary of each triangle. Thus,

$$\tau_K|_e \in \mathcal{P}_0(e) \quad \forall e \in \mathcal{E}(K), \quad \forall K \in \mathcal{T}_h, \qquad \text{and} \qquad \tau_K \geqslant 0.$$

Here $\mathcal{E}(K) = \{e_1^K, \ldots, e_4^K\}$ is the ordered set of faces of K. The function τ is not single-valued in internal faces. We demand that for each K, the function τ_K cannot vanish identically, that is, there exists $e \in \mathcal{E}(K)$ such that $\tau_K|_e > 0$. The HDG method works separately each of the equations in (1.1). There are three unknowns: $(\boldsymbol{q}_h, u_h, \widehat{u}_h) \in \boldsymbol{V}_h \times W_h \times M_h$. Locally, we will think of $(\boldsymbol{q}_K, u_K) \in \mathcal{P}_k(K)^3 \times \mathcal{P}_k(K)$, while $\widehat{u}_h$ will be counted by global face numbering, so we will sometimes refer to values $\widehat{u}_e$. The PDE (1.1a)-(1.1b) is discretized element by element with the equations

$$(\kappa^{-1}\boldsymbol{q}_K, \boldsymbol{r})_K - (u_K, \nabla \cdot \boldsymbol{r})_K + \langle \widehat{u}_h, \boldsymbol{r} \cdot \boldsymbol{\nu}_K \rangle_{\partial K} = 0 \qquad \forall \boldsymbol{r} \in \mathcal{P}_k(K)^3, \quad (1.2a)$$

$$(\nabla \cdot \boldsymbol{q}_K, w)_K + (c\,u_K, w)_K + \langle \tau_K(u_K - \widehat{u}_h), w \rangle_{\partial K} = (f, w)_K \qquad \forall w \in \mathcal{P}_k(K), \quad (1.2b)$$

3

for all $K \in \mathcal{T}_h$. All the remaining equations are defined (and counted) on edges. We first have flux equilibrium on internal faces: for all $\mathcal{E}_h^{\text{int}} \ni e = K \cap \widetilde{K}$, we impose

$$\langle \boldsymbol{q}_K \cdot \boldsymbol{\nu}_K + \tau_K(u_K - \widehat{u}_e), \widehat{v} \rangle_e + \langle \boldsymbol{q}_{\widetilde{K}} \cdot \boldsymbol{\nu}_{\widetilde{K}} + \tau_{\widetilde{K}}(u_{\widetilde{K}} - \widehat{u}_e), \widehat{v} \rangle_e = 0 \quad \forall \widehat{v} \in \mathcal{P}_k(e). \quad (1.2\text{c})$$

This means that the normal numerical flux, $\Phi_K := -\boldsymbol{q}_K \cdot \boldsymbol{\nu}_K - \tau_K(u_K - \widehat{u}_e)$, which is an element of $\mathcal{P}_k(e)$ for all $e \in \mathcal{E}(K)$, is essentially single valued on internal faces, that is $\Phi_K + \Phi_{\widetilde{K}} = 0$ on $e = K \cap \widetilde{K}$. We finally impose the boundary conditions on Dirichlet faces $e \in \mathcal{E}_h^D$

$$\langle \widehat{u}_e, \widehat{v} \rangle_e = \langle u_D, \widehat{v} \rangle_e \qquad \forall \widehat{v} \in \mathcal{P}_k(e), \quad (1.2\text{d})$$

and on Neumann faces $e \in \mathcal{E}_h^N$,

$$-\langle \boldsymbol{q}_K \cdot \boldsymbol{\nu}_K + \tau_K(u_K - \widehat{u}_e), \widehat{v} \rangle_e = \langle \boldsymbol{g}_N \cdot \boldsymbol{\nu}_K, \widehat{v} \rangle_e \qquad \forall \widehat{v} \in \mathcal{P}_k(e), \text{ where } e \in \mathcal{E}(K). \quad (1.2\text{e})$$

Equations (1.2) make up for a square system of linear equations. The hybridization of the methods is a static condensation (substructuring) strategy that allows to write the method as a system of equations where only $\widehat{u}_h$ appears as an unknown. For more methods that fit in this framework, see [4]. Theory has been developed in a series of papers, but revisited and deeply reorganized in [5]. Readers acquainted with programming mixed finite element methods will recognized the hybridized form of [1]. (We will come back to this at the very end of the paper.)

2 Geometric structures

Tetrahedra. All elements will be mapped from the reference tetrahedron $\widehat{K}$ with vertices

$$\widehat{\mathbf{v}}_1 := (0,0,0) \qquad \widehat{\mathbf{v}}_2 := (1,0,0), \qquad \widehat{\mathbf{v}}_3 := (0,1,0), \qquad \widehat{\mathbf{v}}_4 := (0,0,1).$$

Note that $|\widehat{K}| := \text{vol}\,\widehat{K} = 1/6$. Given a tetrahedron with vertices $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ (the order is relevant), we consider the affine mapping $F_K : \widehat{K} \to K$

$$F_K(\widehat{\mathbf{x}}) = \text{B}_K \widehat{\mathbf{x}} + \mathbf{v}_1, \qquad \text{B}_K = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{bmatrix}. \quad (2.1)$$

This map satisfies $F_K(\widehat{\mathbf{v}}_i) = \mathbf{v}_i$, $i \in \{1, 2, 3, 4\}$. All elements of the triangulation will be given with positive orientation, that is,

$$\det \text{B}_K = \Big((\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)\Big) \cdot (\mathbf{v}_4 - \mathbf{v}_1) > 0.$$

In this case $\det \text{B}_K = 6\,|K|$.

Faces. A triangle e in $\mathbb{R}^3$ with vertices $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ (the order is relevant), will be parametrized with

$$\boldsymbol{\phi}_e(s,t) := s\,(\mathbf{w}_2 - \mathbf{w}_1) + t\,(\mathbf{w}_3 - \mathbf{w}_1) + \mathbf{w}_1, \quad \boldsymbol{\phi}_e : \widehat{K}_2 := \{(s,t) \,:\, s, t \geqslant 0, s + t \leqslant 1\} \to e.$$

We note that $|\partial_s \boldsymbol{\phi}_e \times \partial_t \boldsymbol{\phi}_e| = 2|e|$, where $|e|$ is the area of e. The local orientation of the vertices of e gives an orientation to the normal vector. We will define the normal vector so that its norm is proportional to the area of e, that is

$$\mathbf{n}_e := \tfrac{1}{2}\Big((\mathbf{w}_2 - \mathbf{w}_1) \times (\mathbf{w}_3 - \mathbf{w}_1)\Big).$$

Also, if $\widehat{\mathbf{w}}_1 := (0,0)$, $\widehat{\mathbf{w}}_2 := (1,0)$, $\widehat{\mathbf{w}}_3 := (0,1)$, then $\boldsymbol{\phi}_e(\widehat{\mathbf{w}}_i) = \mathbf{w}_i$, for $i \in \{1,2,3\}$.

Boundaries of the tetrahedra. Given a tetrahedron K with vertices $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ we will consider its four faces given in the following order (and with the inherited orientations):

$$
\begin{array}{ccc}
e_1^K & \longleftrightarrow & (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \\
e_2^K & \longleftrightarrow & (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_4) \\
e_3^K & \longleftrightarrow & (\mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_4) \\
e_4^K & \longleftrightarrow & (\mathbf{v}_4, \mathbf{v}_2, \mathbf{v}_3).
\end{array}
\qquad
\begin{bmatrix}
1 & 2 & 3 \\
1 & 2 & 4 \\
1 & 3 & 4 \\
4 & 2 & 3
\end{bmatrix}
\tag{2.2}
$$

(Note that with this orientation of the faces, the normals of the second and fourth faces point outwards, while those of the first and third faces point inwards. This numbering is done for the sake of parametrization.) For integration purposes on ∂K, we will use parametrizations of the faces $e_\ell^K \in \mathcal{E}(K)$

$$\boldsymbol{\phi}_\ell^K : \widehat{K}_2 \to e_\ell^K \qquad \ell \in \{1,2,3,4\},$$

given by the formulas

$$
\begin{aligned}
\boldsymbol{\phi}_1^K(s,t) &:= F_K(s,t,0), \\
\boldsymbol{\phi}_2^K(s,t) &:= F_K(s,0,t), \\
\boldsymbol{\phi}_3^K(s,t) &:= F_K(0,s,t), \\
\boldsymbol{\phi}_4^K(s,t) &:= F_K(s,t,1-s-t).
\end{aligned}
\tag{2.3}
$$

Consider the affine invertible maps $F_\mu : \widehat{K}_2 \to \widehat{K}_2$ given by the formulas

$$
\begin{aligned}
F_1(s,t) &:= (s,t) \\
F_2(s,t) &:= (t,s) \\
F_3(s,t) &:= (t, 1-s-t) \\
F_4(s,t) &:= (s, 1-s-t) \\
F_5(s,t) &:= (1-s-t, s) \\
F_6(s,t) &:= (1-s-t, t)
\end{aligned}
\qquad
\begin{bmatrix}
\mathbf{1} & \mathbf{2} & \mathbf{3} \\
\mathbf{1} & 3 & 2 \\
3 & 1 & 2 \\
3 & \mathbf{2} & 1 \\
2 & 3 & 1 \\
2 & 1 & \mathbf{3}
\end{bmatrix}.
\tag{2.4}
$$

The table on the right shows the indices of the images of the vertices $(\widehat{\mathbf{w}}_1, \widehat{\mathbf{w}}_2, \widehat{\mathbf{w}}_3)$, with boldface font for those that stay fixed. We note that F_2, F_4 and F_6 change orientation. Take now a tetrahedron K, and assume that $e = e_\ell^K$, i.e., e_ℓ^K is the face e in a global list of faces. We thus have six possible cases of how the parametrizations $\boldsymbol{\phi}_\ell^K$ and $\boldsymbol{\phi}_e$ match. We will encode this information in a matrix $\mathrm{perm}(K, \ell)$ so that

$$\boldsymbol{\phi}_e \circ F_\mu = \boldsymbol{\phi}_\ell^K, \text{ if } e = e_\ell^K \text{ and } \mu = \mathrm{perm}(K, \ell). \tag{2.5}$$

We will refer to this matrix as the permutation matrix.

Data structure. The basic tetrahedrization of Ω (including information on Dirichlet and Neumann boundaries) is given through four fields of a data structure `T`:

- `T.coordinates` is an $N_{\mathrm{ver}} \times 3$ matrix with the coordinates of the vertices of the triangulation.

- `T.elements` is an $N_{\mathrm{elt}} \times 4$ matrix, whose K-th row of the matrix contains the indices of the vertices of K.

- `T.dirichlet` is an $N_{\mathrm{dir}} \times 3$ matrix, with the vertex numbers for the Dirichlet faces.

- `T.neumann` is an $N_{\mathrm{neu}} \times 3$ matrix, with the vertex numbers for the Neumann faces.

Positive orientation, of listings of vertices for elements and faces, is always assumed, In *expanded form*, the tetrahedral data structure contains many more useful fields. All these elements can be easily precomputed. It will be useful for what follows to assume that they are easy to access whenever needed.

- `T.faces` is an $N_{\mathrm{fc}} \times 4$ matrix with a list of faces: the first three columns contain the global vertex numbers for the faces (its order will give the intrinsic parametrization of the face); Dirichlet and Neumann faces are numbered exacly as in `T.dirichlet` and `T.neumann`, the fourth column contains an index:

    - 0 for interior faces
    - 1 for Dirichlet faces
    - 2 for Neumann faces

- `T.dirfaces` and `T.neufaces` are row vectors with the list of Dirichlet and Neumann faces, that is, they point out what rows of `T.faces` contain a 1 (resp a 2) in the last column.

- `T.facebyele` is an $N_{\mathrm{elt}} \times 4$ matrix, whose K-th row contains the numbers of faces that make up ∂K, with the faces given in the order shown in the table in (2.2). Note that this is the matrix we have described as e_ℓ^K.

- `T.perm` is an $N_{\mathrm{elt}} \times 4$ matrix containing numbers from 1 to 6. Its K-th row indicates what permutations are needed for each of the faces to get to the proper numbering of the face, i.e., this is just the matrix $\mathrm{perm}(K, \ell)$.

- `T.volume` is an $N_{\mathrm{elt}} \times 1$ column vector with the volumes of the elements.

- `T.area` is an $N_{\mathrm{fc}} \times 1$ column vector with the areas of the faces.

- `T.normals` is an $N_{\mathrm{elt}} \times 12$ matrix with the *non-normalized* normal vectors for the faces of the elements; its K-th row contains four row vectors of three components each

$$\left[\begin{array}{c|c|c|c} \mathbf{n}_1^\top & \mathbf{n}_2^\top & \mathbf{n}_3^\top & \mathbf{n}_4^\top \end{array} \right]$$

so that $\mathbf{n}_\ell$ is the normal vector to the face e_ℓ^K, *pointing outwards* and such that $|\mathbf{n}_\ell| = |e_\ell^K|$.

3 Volume integrals

Pseudo-matlab notation. In order to exploit the vectorization capabilities of Matlab, we will use the following notation to describe some particular operations. First of all, for a function $f(x, y, z)$, we will automatically assume that it is vectorized and can thus be simultaneously evaluated in many points stored in equally sizes matrix, so that $f(\mathrm{X}, \mathrm{Y}, \mathrm{Z})$ is a matrix with the same size as X, Y, and Z. Our default will be that vectors are column vectors. Whenever the sizes of a column vector $\mathbf{u}$ and a matrix A are compatible, we will write $\mathbf{u}^\top \odot \mathrm{A} := \mathrm{A}\,\mathrm{diag}(\mathbf{u})$, and $\mathbf{u} \odot \mathrm{A} := \mathrm{diag}(\mathbf{u})\,\mathrm{A}$. At the entry level, these are the operations

$$(\mathbf{u}^\top \odot \mathrm{A})_{ij} = u_j \mathrm{A}_{ij} \qquad \text{and} \qquad (\mathbf{u} \odot \mathrm{A})_{ij} = u_i \mathrm{A}_{ij},$$

which can be easily performed using Matlab's `bsxfun` utility. Also, the Kronecker product will be used in the following particular situation:

$$\mathbf{c}^\top \otimes \mathrm{A} = \left[\begin{array}{c|c|c|c} c_1 \mathrm{A} & c_2 \mathrm{A} & \cdots & c_N \mathrm{A} \end{array} \right].$$

Finally, given a matrix A, $\mathbf{a}_i^\top := \mathrm{row}(\mathrm{A}, i)$ will be used to denote the row vector corresponding to the i-th row of A.

Three dimensional quadrature. To compute or approximate element integrals we will consider a *quadrature formula* on the reference element $\widehat{K}$:

$$\int_{\widehat{K}} \widehat{\phi} \approx \frac{1}{6} \sum_{q=1}^{N_{\mathrm{qd}}} \widehat{\omega}_q \widehat{\phi}(\widehat{\mathbf{p}}_q), \qquad \widehat{\mathbf{p}}_q := (\widehat{x}_q, \widehat{y}_q, \widehat{z}_q), \qquad \sum_{q=1}^{N_{\mathrm{qd}}} \widehat{\omega}_q = 1. \tag{3.1}$$

Such formulas can be easily found in the literature [7, 11]. We will find it convenient to store the quadrature points with their barycentric coordinates $(1 - \widehat{x}_q - \widehat{y}_q - \widehat{z}_q, \widehat{x}_q, \widehat{y}_q, \widehat{z}_q)$ in a $N_{\mathrm{qd}} \times 4$ matrix Λ (rows correspond to quadrature points). To integrate on a general element we use the mapping $F_K : \widehat{K} \to K$ of (2.1) and proceed as follows

$$\int_K \phi = \det \mathrm{B}_K \int_{\widehat{K}} \phi \circ F_K \approx |K| \sum_{q=1}^{N_{\mathrm{qd}}} \widehat{\omega}_q \phi(\mathbf{p}_q^K), \qquad \mathbf{p}_q^K := F_K(\widehat{\mathbf{p}}_q). \tag{3.2}$$

Piecewise polynomials. The bases of the local polynomial spaces $\mathcal{P}_3(K)$, containing $d_3 = d_3(k) := \binom{k+3}{3}$ elements, will be obtained by pushing forward a basis in the reference element. For this we will use the three dimensional Dubiner basis [8], which is given in the enlarged element $2\widehat{K} - (1,1,1)^\top$, where it is L^2-orthogonal. The Dubiner basis is evaluated using a Duffy-type transformation and Jacobi polynomials. What is needed for HDG is the evaluation of the Dubiner basis $\{\check{P}_i\}$ and of its partial derivatives $\partial_\alpha \check{P}_i$. *We will assume that the basis is ordered in hierarchical form*, that is, polynomials of degree k are stored after all polynomials of degree $k-1$ for all k. We then consider the local bases $\{P_i^K\}$ given by the relations

$$P_i^K \circ F_K = \widehat{P}_i := \check{P}_i(2 \cdot -(1,1,1)^\top). \tag{3.3}$$

An element of the space W_h will be usually stored as a $d_k \times N_{\mathrm{elt}}$ matrix, where each column contains the coefficients of the local polynomial function in the local basis.

Source terms. We first extract all nodal information of the grid in three $4 \times N_{\mathrm{elt}}$ matrices $\mathrm{X}^{\mathcal{T}}$, $\mathrm{Y}^{\mathcal{T}}$, $\mathrm{Z}^{\mathcal{T}}$. For instance, the element $\mathrm{X}_{i,K}^{\mathcal{T}}$ contains the x coordinate of the i-th node of element K. If Λ is the $N_{\mathrm{qd}} \times 4$ matrix with the barycentric coordinates of the quadrature points in $\widehat{K}$, then

$$\mathrm{X} := \Lambda\mathrm{X}^{\mathcal{T}}, \qquad \mathrm{Y} := \Lambda\mathrm{Y}^{\mathcal{T}}, \qquad \mathrm{Z} := \Lambda\mathrm{Z}^{\mathcal{T}} \tag{3.4}$$

are $N_{\mathrm{qd}} \times N_{\mathrm{elt}}$ matrices with the coordinates of all quadrature points. Further, let us consider the $N_{\mathrm{qd}} \times d_3$ matrix

$$\mathrm{P}_{qj} := \widehat{P}_j(\widehat{\mathbf{p}}_q), \qquad q = 1, \ldots, N_{\mathrm{qd}}, \qquad j = 1, \ldots, d_3. \tag{3.5}$$

The computational representation of the formula (see (3.2) and (3.3))

$$\int_K f P_i^K = 6|K| \int_{\widehat{K}} (f \circ F_K)\, \widehat{P}_i \approx |K| \sum_{q=1}^{N_{\mathrm{qd}}} f(\mathbf{p}_q^K)\widehat{\omega}_q \widehat{P}_i(\widehat{\mathbf{p}}_q), \qquad i = 1, \ldots, d_3, \quad K \in \mathcal{T}_h,$$

is given by (see (3.4) and (3.5))

$$\mathbf{vol}^\top \odot \big((\widehat{\boldsymbol{\omega}} \odot \mathrm{P})^\top f(\mathrm{X}, \mathrm{Y}, \mathrm{Z})\big),$$

where $\mathbf{vol}$ is the column vector containing the volumes of all elements and $\widehat{\boldsymbol{\omega}}$ is a column vector with the weights of the quadrature rule.

Mass matrices. In order to compute mass matrices with variable density function m, we use (3.2)-(3.3) and write

$$\int_K m\, P_i^K P_j^K \approx |K| \sum_{q=1}^{N_{\mathrm{qd}}} m(\mathbf{p}_q^K)\big(\widehat{\omega}_q \widehat{P}_i(\widehat{\mathbf{p}}_q)\widehat{P}_j(\widehat{\mathbf{p}}_q)\big), \qquad i, j, = 1, \ldots, d_3, \quad K \in \mathcal{T}_h.$$

We then evaluate the density at all quadrature points (3.4) to get an $N_{\mathrm{qd}} \times N_{\mathrm{elt}}$ matrix, already weighted by the element volumes,

$$\mathrm{M} = \mathbf{vol}^\top \odot m(\mathrm{X}, \mathrm{Y}, \mathrm{Z}), \tag{3.6a}$$

and finally loop over quadrature nodes using the rows of (3.5)

$$\sum_{q=1}^{N_{\mathrm{qd}}} \mathbf{m}_q^\top \otimes (\widehat{\omega}_q \mathbf{p}_q \, \mathbf{p}_q^\top), \qquad \mathbf{m}_q^\top = \mathrm{row}(\mathrm{M}, q), \quad \mathbf{p}_q^\top = \mathrm{row}(\mathrm{P}, q). \tag{3.6b}$$

The result comes out as a $d_3 \times (d_3 N_{\mathrm{elt}})$ matrix that can be easily reshaped to a $d_3 \times d_3 \times N_{\mathrm{elt}}$ array.

Convection matrices. We start by computing three $d_3 \times d_3$ matrices in the reference element

$$\widehat{\mathrm{C}}_{ij}^\star := \int_{\widehat{K}} \widehat{P}_i \partial_{\widehat{\star}} \widehat{P}_j = \tfrac{1}{6} \sum_{q=1}^{N_{\mathrm{qd}}} \widehat{\omega}_q \widehat{P}_i(\widehat{\mathbf{p}}_q) \partial_{\widehat{\star}} \widehat{P}_j(\widehat{\mathbf{p}}_q), \qquad i, j = 1, \ldots, d_3, \quad \star \in \{x, y, z\}. \tag{3.7}$$

(We assume that the quadrature rule is of sufficiently high order to compute these matrices exactly.) To do this, we require the matrix P in (3.5) plus three matrices with derivatives of the basis functions in the reference element

$$\mathrm{P}_{qi}^\star := (\partial_{\widehat{\star}} \widehat{P}_i)(\widehat{\mathbf{p}}_q) \qquad q = 1, \ldots, N_{\mathrm{qd}}, \quad i = 1, \ldots, d_3, \quad \star \in \{x, y, z\}. \tag{3.8}$$

Then,

$$\widehat{\mathrm{C}}^\star = \tfrac{1}{6} \mathrm{P}^\top \mathrm{diag}(\widehat{\boldsymbol{\omega}}) \, \mathrm{P}^\star = \tfrac{1}{6} (\widehat{\boldsymbol{\omega}} \odot \mathrm{P})^\top \mathrm{P}^\star. \tag{3.9}$$

Next, we deal with the elements of the associated Piola transform. The elements of the 3×3 matrices

$$\det \mathrm{B}_K \mathrm{B}_K^{-\top} = \begin{bmatrix} a_{xx}^K & a_{xy}^K & a_{xz}^K \\ a_{yx}^K & a_{yy}^K & a_{yz}^K \\ a_{zx}^K & a_{zy}^K & a_{zz}^K \end{bmatrix}, \qquad K \in \mathcal{T}_h, \tag{3.10}$$

can be computed using the coordinates of the vertices counted by elements (these are the rows of the matrices $\mathrm{X}^{\mathcal{T}}$, $\mathrm{Y}^{\mathcal{T}}$, and $\mathrm{Z}^{\mathcal{T}}$) using the formulas:

$$\begin{aligned} a_{xx} &= (y_3 - y_1)(z_4 - z_1) - (y_4 - y_1)(z_3 - z_1), \\ a_{xy} &= (y_4 - y_1)(z_2 - z_1) - (y_2 - y_1)(z_4 - z_1), \\ a_{xz} &= (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1), \\ a_{yx} &= (x_4 - x_1)(z_3 - z_1) - (x_3 - x_1)(z_4 - z_1), \\ a_{yy} &= (x_2 - x_1)(z_4 - z_1) - (x_4 - x_1)(z_2 - z_1), \\ a_{yz} &= (x_3 - x_1)(z_2 - z_1) - (x_2 - x_1)(z_3 - z_1), \\ a_{zx} &= (x_3 - x_1)(y_4 - y_1) - (x_4 - x_1)(y_3 - y_1), \\ a_{zy} &= (x_4 - x_1)(y_2 - y_1) - (x_2 - x_1)(y_4 - y_1), \\ a_{zz} &= (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1). \end{aligned}$$

(Reference to K has been dropped to simplify the expression.) A simple change of variables leads to

$$\int_K P_i^K \partial_\star P_j^K = \sum_{\#\in\{x,y,z\}} a_{\star\#}^K \int_{\widehat{K}} \widehat{P}_i \partial_{\widehat{\#}} \widehat{P}_j = \sum_{\#\in\{x,y,z\}} a_{\star\#}^K \widehat{\mathsf{C}}_{ij}^\#,$$

which, using the matrices (3.9), can be implemented with Kronecker products

$$\sum_{\#\in\{x,y,z\}} \mathbf{a}_{\star\#}^\top \otimes \widehat{\mathsf{C}}^\#, \qquad \star \in \{x, y, z\}. \tag{3.11}$$

The result are three $d_3 \times (d_3 N_{\mathrm{elt}}) \equiv d_3 \times d_3 \times N_{\mathrm{elt}}$ matrices.

4 Surface integrals

Integrals on faces. Two dimensional quadrature rules will be given in the reference element $\widehat{K}_2$, using points and weights so that

$$\int_{\widehat{K}_2} \widehat{\phi} \approx \tfrac{1}{2} \sum_{r=1}^{N_{\mathrm{qd2}}} \varpi_r \widehat{\phi}(\widehat{\mathbf{q}}_r), \qquad \widehat{\mathbf{q}}_r = (\widehat{s}_r, \widehat{t}_r), \qquad \sum_{r=1}^{N_{\mathrm{qd2}}} \varpi_r = 1.$$

To compute an integral on $e \in \mathcal{E}_h$, we simply parametrize from $\widehat{K}_2$ and proceed accordingly:

$$\int_e \phi = 2|e| \int_{\widehat{K}_2} \phi \circ \boldsymbol{\phi}_e \approx |e| \sum_{r=1}^{N_{\mathrm{qd2}}} \varpi_r \phi(\mathbf{q}_r^e) \qquad \text{with} \qquad \mathbf{q}_r^e := \boldsymbol{\phi}_e(\widehat{\mathbf{q}}_r).$$

For practical purposes, we will keep the barycentric coordinates of the quadrature points $(1 - s_r - t_r, s_r, t_r)$ in an $N_{\mathrm{qd2}} \times 3$ matrix Ξ.

Integrals on boundaries of tetrahedra. In many cases we will be integrating on a face that is given with geometric information of an adjacent tetrahedron. The quadrature points $\widehat{\mathbf{q}}_r$ lead to four groups of quadrature points on the faces of $\widehat{K}$ (see (2.3)):

$$\widehat{\mathbf{q}}_r^1 := (\widehat{s}_r, \widehat{t}_r, 0),$$
$$\widehat{\mathbf{q}}_r^2 := (\widehat{s}_r, 0, \widehat{t}_r),$$
$$\widehat{\mathbf{q}}_r^3 := (0, \widehat{s}_r, \widehat{t}_r),$$
$$\widehat{\mathbf{q}}_r^4 := (\widehat{s}_r, \widehat{t}_r, 1 - \widehat{s}_r - \widehat{t}_r),$$

For a given $\psi : K \to \mathbb{R}$, we can approximate (see (2.4) and (2.5))

$$\int_{e_\ell^K} \psi \approx |e_\ell^K| \sum_{r=1}^{N_{\mathrm{qd2}}} \varpi_r \psi(\mathbf{q}_{r,\ell}^K), \;\; \text{with } \mathbf{q}_{r,\ell}^K := \mathrm{F}_K(\widehat{\mathbf{q}}_r^\ell) = \boldsymbol{\phi}_{e_\ell^K}(F_{\mathrm{perm}(K,\ell)}(\widehat{\mathbf{q}}_r)), \tag{4.1}$$

and thus

$$\int_{\partial K} \psi \approx \sum_{\ell=1}^{4} |e_\ell^K| \sum_{r=1}^{N_{\mathrm{qd2}}} \varpi_r \psi(\mathbf{q}_{r,\ell}^K). \tag{4.2}$$

Note that the use of the permutation index $\mathrm{perm}(K,\ell)$ in (4.1) factors out the natural parametrization of $e = e_\ell^K$ on the left, which will be necessary for functions on e that are defined by pushing forward functions on $\widehat{K}_2$, that is, for functions ψ such that we can evaluate $\psi \circ \boldsymbol{\phi}_e$.

Bases, faces, and boundaries. Our starting point is the Dubiner basis $\{\check{D}_i\}$ [8], which is orthogonal in the enlarged element $2\widehat{K}_2 - (1,1)^\top$. We assume it to be given in hierarchical form. The elements of M_h will be described via their coefficients in the basis D_i^e, where

$$D_i^e \circ \boldsymbol{\phi}_e = \widehat{D}_i = \check{D}_i(2 \cdot -(1,1)^\top), \qquad i = 1,\ldots,d_2, \quad e \in \mathcal{E}_h, \quad d_2 = d_2(k) := \binom{k+2}{2},$$

so that they are stored in form of a $d_2 \times N_{\mathrm{fc}}$ matrix (recall that $N_{\mathrm{fc}} = \#\mathcal{E}_h$).

Types of boundary integrals. There will be three different kinds of integrals on ∂K: (a) products of traces of polynomials on K, (b) products of piecewise polynomials defined on ∂K, (c) products of traces of polynomials of K by piecewise polynomials on ∂K. Each of these integrals will involve some kind of piecewise constant weight function. Piecewise constant functions on the boundaries of the elements (with different values on internal faces) will be described with $4 \times N_{\mathrm{fc}}$ matrices. We will be using four examples of this kind of functions:

$$\begin{aligned} \mathrm{T}_\ell^K &:= \tau_\ell^K |e_\ell^K|, \\ \mathrm{N}_{\ell,K}^\star &:= n_{\ell,\star}^K = \nu_{\ell,\star}^K |e_\ell^K|, \end{aligned} \qquad \ell = 1,\ldots,4, \quad K \in \mathcal{T}_h, \quad \star \in \{x,y,z\}. \tag{4.3}$$

Here $\mathbf{n}_\ell^K = (n_{\ell,x}^K, n_{\ell,y}^K, n_{\ell,z}^K)$ is the normal vector on the ℓ-th face of K, with the normalization $|\mathbf{n}_\ell^K| = |e_\ell^K|$ (see Section 2). The information of these four piecewise constant functions is readily available in the enhanced geometric data structure.

Type (a) matrices. We can compute the integrals

$$\int_{\partial K} \tau_K P_i^K P_j^K = \sum_{\ell=1}^{4} \mathrm{T}_\ell^K \left(\sum_{r=1}^{N_{\mathrm{qd2}}} \widehat{P}_i(\widehat{\mathbf{q}}_r^\ell) \varpi_r \widehat{P}_j(\widehat{\mathbf{q}}_r^\ell) \right) \qquad i,j = 1,\ldots,d_3, \quad K \in \mathcal{T}_h, \tag{4.4}$$

using a sufficiently precise quadrature rule. If we consider the matrices

$$\mathrm{P}_{ri}^\ell := \widehat{P}_i(\widehat{\mathbf{q}}_r^\ell), \qquad r = 1,\ldots,N_{\mathrm{qd2}}, \quad i = 1,\ldots,d_3, \qquad \ell \in \{1,2,3,4\}, \tag{4.5}$$

then (4.4) can be computed as

$$\sum_{\ell=1}^{4} \mathbf{t}_\ell^\top \otimes \left((\boldsymbol{\varpi} \odot \mathrm{P}^\ell)^\top \mathrm{P}^\ell \right), \qquad \mathbf{t}_\ell^\top = \mathrm{row}(\mathrm{T},\ell) \tag{4.6}$$

using the matrix T in (4.3). The result is a $d_3 \times (d_3 N_{\mathrm{elt}}) \equiv d_3 \times d_3 \times N_{\mathrm{elt}}$ array.

Type (b) matrices. To compute the matrices

$$\tau_\ell^K \int_{e_\ell^K} D_i^{e_\ell^K} D_j^{e_\ell^K} = \mathrm{T}_\ell^K \sum_{r=1}^{N_{\mathrm{qd2}}} \varpi_r \widehat{D}_i(\widehat{\mathbf{q}}_r)\widehat{D}_j(\widehat{\mathbf{q}}_r), \qquad \begin{aligned} &i,j = 1,\ldots,d_2,\\ &K \in \mathcal{T}_h, \quad \ell \in \{1,2,3,4\}, \end{aligned}$$

we compute the matrices

$$\mathrm{D}_{ri} := \widehat{D}_i(\widehat{\mathbf{q}}_r) \qquad r = 1,\ldots,N_{\mathrm{qd2}}, \quad i = 1,\ldots,d_2 \tag{4.7}$$

and mix them in the form

$$\mathbf{t}_\ell^\top \otimes \big((\boldsymbol{\varpi} \odot \mathrm{D})^\top \mathrm{D}\big), \qquad \mathbf{t}_\ell^\top = \mathrm{row}(\mathrm{T},\ell). \tag{4.8}$$

A simpler option is taking advantage of the fact that the Dubiner basis is orthogonal, so these computations yield diagonal matrices. The result are four $d_2 \times (d_2 N_{\mathrm{elt}}) \equiv d_2 \times d_2 \times N_{\mathrm{elt}}$ matrices. They will be the diagonal blocks of a $(4d_2) \times (4d_2) \times N_{\mathrm{elt}}$ matrix that will be used in the local solvers.

Type (c) matrices. Let $\boldsymbol{\xi}$ be a piecewise constant function on the set of boundaries of the elements (in practice, one of the functions described in (4.3)). Let $\boldsymbol{\xi}_\mu$ be the piecewise constant functions given by

$$\xi_{\ell,\mu}^K := \xi_\ell^K \mathbf{1}_{\mathrm{perm}(K,\ell)=\mu}, \qquad \ell \in \{1,2,3,4\}, \qquad K \in \mathcal{T}_h, \qquad \mu \in \{1,2,3,4,5,6\}. \tag{4.9}$$

Following (4.1), we can then compute

$$\begin{aligned}
\frac{\xi_\ell^K}{|e_\ell^K|} \int_{e_\ell^K} D_i^{e_\ell^K} P_j^K &= \xi_\ell^K \sum_{r=1}^{N_{\mathrm{qd2}}} \widehat{D}_i(F_{\mathrm{perm}(K,\ell)}(\widehat{\mathbf{q}}_r))\varpi_r \widehat{P}_j(\widehat{\mathbf{q}}_r^\ell)\\
&= \sum_{\mu=1}^{6} \xi_{\ell,\mu}^K \Big(\sum_{r=1}^{N_{\mathrm{qd2}}} \widehat{D}_i(F_\mu(\widehat{\mathbf{q}}_r)) \varpi_r \widehat{P}_j(\widehat{\mathbf{q}}_r^\ell) \Big),
\end{aligned}$$

for $i = 1,\ldots,d_2$, $j = 1,\ldots,d_3$, $K \in \mathcal{T}_h$ and $\ell \in \{1,2,3,4\}$. Using (4.5), the matrices

$$\mathrm{D}_{ri}^\mu := \widehat{D}_i(F_\mu(\widehat{\mathbf{q}}_r)) \qquad r = 1,\ldots,N_{\mathrm{qd2}}, \quad i = 1,\ldots,d_2, \quad \mu \in \{1,\ldots,6\},$$

and (4.9), the previous computation reduces to

$$\sum_{\mu=1}^{6} \boldsymbol{\xi}_{\ell,\mu}^\top \otimes \big((\boldsymbol{\varpi} \odot \mathrm{D}^\mu)^\top \mathrm{P}^\ell\big), \qquad \boldsymbol{\xi}_{\ell,\mu}^\top = \mathrm{row}(\boldsymbol{\xi}_\mu,\ell) \qquad \ell = 1,2,3,4. \tag{4.10}$$

The result is four $d_2 \times (d_3 N_{\mathrm{elt}}) \equiv d_2 \times d_3 \times N_{\mathrm{elt}}$ matrices that are stored as a single $(4d_2) \times d_3 \times N_{\mathrm{elt}}$ array, by stacking the blocks for $\ell = 1,2,3,4$ on top of each other ($\ell = 1$ on top).

5 Local solvers

The local solvers that we next define are related to the pair of discrete equations (1.2a)-(1.2b).

Matrices and bilinear forms. In order to recognize the matrices that we have computed with terms in the bilinear forms of the HDG method, we need some notation. We consider the space

$$\mathcal{R}_k(\partial K) := \prod_{e \in \mathcal{E}(K)} \mathcal{P}_k(e), \qquad \dim \mathcal{R}_k(\partial K) = 4d_2.$$

The degrees of freedom for this last space are organized by taking one face at a time in the order they are given by `T.facebyele`. For (non-symmetric) bilinear forms we will use the convention that the bilinear form $b(u,v)$ is related to the matrix $b(U_j, V_i)$, where $\{U_j\}$ is a basis of the space of u and $\{V_i\}$ is a basis of the space for v. This is equivalent to saying that the unknown will always be placed as the left-most argument in the bilinear form and the test function will occupy the right-most location.

Volume terms. We start by computing mass matrices associated to two functions (κ^{-1} and c), and the three convection matrices:

$$\mathrm{M}_{\kappa^{-1}}^K, \qquad \mathrm{M}_c^K, \qquad \mathrm{C}_x^K, \qquad \mathrm{C}_y^K, \qquad \mathrm{C}_z^K,$$

where (see (3.6) and (3.11))

$$(\mathrm{M}_m^K)_{ij} = \int_K m P_i^K P_j^K, \qquad (\mathrm{C}_\star^K)_{ij} = \int_K P_i^K \partial_\star P_j^K.$$

Each of these matrices is $d_3 \times d_3 \times N_{\mathrm{elt}}$. They correspond to the bilinear forms

$$(m\, u_h, v_h)_K, \qquad (\partial_\star u_h, v_h)_K, \qquad u_h, v_h \in \mathcal{P}_k(K).$$

Surface terms. We next compute all matrices related to integrals on interfaces:

$$\tau \mathrm{PP}^K, \qquad \tau \mathrm{DP}^K, \qquad n_x \mathrm{DP}^K, \qquad n_y \mathrm{DP}^K, \qquad n_z \mathrm{DP}^K, \qquad \tau \mathrm{DD}^K.$$

The first of these arrays is $d_3 \times d_3 \times N_{\mathrm{elt}}$, the next four are $4d_2 \times d_3 \times N_{\mathrm{elt}}$ and the last one is $4d_2 \times 4d_2 \times N_{\mathrm{elt}}$. The first matrix and associated bilinear form (see (4.6)) are

$$\tau \mathrm{PP}_{ij}^K = \int_{\partial K} \tau_K P_i^K P_j^K, \qquad \langle \tau u_h, v_h \rangle_{\partial K}, \qquad u_h, v_h \in \mathcal{P}_k(K).$$

The second one (see (4.10)) corresponds to the bilinear form

$$\langle \tau_K u_h, \widehat{v}_h \rangle_{\partial K}, \qquad u_h \in \mathcal{P}_k(K), \quad \widehat{v}_h \in \mathcal{R}_k(\partial K),$$

or equivalently to $\langle \tau_K u_h, \widehat{v}_h \rangle_e$, for $u_h \in \mathcal{P}_k(K)$, $\widehat{v}_h \in \mathcal{P}_k(e)$, and $e \in \mathcal{E}(K)$. The matrices associated to the components of the normal vector $\boldsymbol{\nu} = (\nu_x, \nu_y, \nu_z)$ (see (4.10) again) are related to the bilinear forms

$$\langle \nu_\star u_h, \widehat{v}_h \rangle_{\partial K}, \qquad u_h \in \mathcal{P}_k(K), \quad \widehat{v}_h \in \mathcal{R}_k(\partial K), \qquad \star \in \{x, y, z\}.$$

The last matrix (see (4.8)) corresponds to

$$\langle \tau \widehat{u}_h, \widehat{v}_h \rangle_{\partial K}, \qquad \widehat{u}_h, \widehat{v}_h \in \mathcal{R}_k(\partial K),$$

and is therefore block diagonal. Finally we compute the vectors of tests of f with the basis elements of $\mathcal{P}_k(K)$: $\mathbf{f}^K \in \mathbb{R}^{d_3}$.

Matrices related to local solvers. The $4d_3 \times 4d_3 \times N_{\text{elt}}$ array and the $4d_3 \times 4d_2 \times N_{\text{elt}}$ array with respective slices

$$
\mathbb{A}_1^K := \begin{bmatrix} \mathrm{M}_{\kappa^{-1}}^K & \mathrm{O} & \mathrm{O} & -(\mathrm{C}_x^K)^\top \\ \mathrm{O} & \mathrm{M}_{\kappa^{-1}}^K & \mathrm{O} & -(\mathrm{C}_y^K)^\top \\ \mathrm{O} & \mathrm{O} & \mathrm{M}_{\kappa^{-1}}^K & -(\mathrm{C}_z^K)^\top \\ \mathrm{C}_x^K & \mathrm{C}_y^K & \mathrm{C}_y^K & \mathrm{M}_c^K + \tau\mathrm{PP}^K \end{bmatrix}, \qquad \mathbb{A}_2^K := \begin{bmatrix} (n_x\mathrm{DP}^K)^\top \\ (n_y\mathrm{DP}^K)^\top \\ (n_z\mathrm{DP}^K)^\top \\ -(\tau\mathrm{DP}^K)^\top \end{bmatrix}, \qquad (5.1)
$$

are the matrix representations of the bilinear forms

$$
a_1^K : \big(\mathcal{P}_k(K)^3 \times \mathcal{P}_k(K)\big) \times \big(\mathcal{P}_k(K)^3 \times \mathcal{P}_k(K)\big) \longrightarrow \mathbb{R},
$$
$$
a_2^K : \mathcal{R}_k(\partial K) \times \big(\mathcal{P}_k(K)^3 \times \mathcal{P}_k(K)\big) \longrightarrow \mathbb{R},
$$

given by

$$
\begin{aligned}
a_1^K((\boldsymbol{q}_h, u_h), (\boldsymbol{r}_h, w_h)) &:= (\kappa_h^{-1}, \boldsymbol{r}_h)_K - (u_h, \nabla \cdot \boldsymbol{r}_h)_K \\
&\quad + (\nabla \cdot \boldsymbol{q}_h, w_h)_K + (c\, u_h, w_h)_K + \langle \tau u_h, v_h \rangle_{\partial K}, \\
a_2^K(\widehat{u}_h, (\boldsymbol{r}_h, w_h)) &:= \langle \widehat{u}_h, \boldsymbol{r}_h \cdot \boldsymbol{\nu} \rangle_{\partial K} - \langle \tau \widehat{u}_h, w_h \rangle_{\partial K}.
\end{aligned}
$$

We also consider the $4d_3 \times N_{\text{elt}}$ matrix with columns

$$
\mathbb{A}_f^K := \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}^K \end{bmatrix}, \qquad (5.2)
$$

If $\widehat{u}_h \in M_h$ is known, we can solve the local problems looking for $\boldsymbol{q}_h \in \boldsymbol{V}_h = W_h^3$ and $u_h \in W_h$, satisfying (1.2a)-(1.2b). Representing $\widehat{u}_h|_{\partial K} \in \mathcal{R}_k(\partial K)$ with a vector $\mathbf{u}_{\partial K} \in \mathbb{R}^{4d_2}$, the matrix representation of this local solution is

$$
\begin{bmatrix} \mathbf{q}_K \\ \mathbf{u}_K \end{bmatrix} = -(\mathbb{A}_1^K)^{-1}\mathbb{A}_2^K \mathbf{u}_{\partial K} + (\mathbb{A}_1^K)^{-1}\mathbb{A}_f^K \in \mathbb{R}^{4d_3}. \qquad (5.3)
$$

Note that once the local matrices have been computed, the construction of the three dimentional arrays (5.1) can be easily carried out by stacking the already created three dimensional arrays.

Flux operators. Consider now the $4d_2 \times 4d_3 \times N_{\text{elt}}$ array with slices

$$
\mathbb{A}_3^K := \begin{bmatrix} n_x\mathrm{DP}^K & n_y\mathrm{DP}^K & n_z\mathrm{DP}^K & \tau\mathrm{DP}^K \end{bmatrix}, \qquad (5.4)
$$

the $4d_2 \times 4d_2 \times N_{\text{elt}}$ array with slices

$$
\mathbb{C}^K := \mathbb{A}_3^K(\mathbb{A}_1^K)^{-1}\mathbb{A}_2^K + \tau\mathrm{DD}^K, \qquad (5.5)
$$

and the $4d_2 \times N_{\text{elt}}$ matrix with columns

$$
\mathbb{C}_f^K := \mathbb{A}_3^K(\mathbb{A}_1^K)^{-1}\mathbb{A}_f^K. \qquad (5.6)
$$

14

The meaning of these matrices can be made clear by looking at boundary fluxes. Given $(\boldsymbol{q}_h, u_h, \widehat{u}_h) \in \boldsymbol{V}_h \times W_h \times M_h$ –satisfying equations (1.2a) and (1.2b)–, the HDG method is based on the construction of the flux function

$$\Phi_K := -\boldsymbol{q}_h \cdot \boldsymbol{\nu} - \tau(u_h - \widehat{u}_h) : \partial K \to \mathbb{R}.$$

Instead of this quantity, we pay attention to how it creates a linear form

$$\mathcal{R}_k(\partial K) \ni \widehat{v}_h \longmapsto -\langle \boldsymbol{q}_h \cdot \boldsymbol{\nu} + \tau(u_h - \widehat{u}_h), \widehat{v}_h \rangle_{\partial K} = -\langle \boldsymbol{q}_h \cdot \boldsymbol{\nu} + \tau u_h, \widehat{v}_h \rangle_{\partial K} + \langle \tau \widehat{u}_h, \widehat{v}_h \rangle_{\partial K},$$

whose matrix representation is

$$
\begin{aligned}
-\mathbb{A}_3^K \begin{bmatrix} \mathbf{q}_K \\ \mathbf{u}_K \end{bmatrix} + \tau \mathrm{DD}^K \mathbf{u}_{\partial K} &= \mathbb{A}_3^K (\mathbb{A}_1^K)^{-1} \mathbb{A}_2^K \mathbf{u}_{\partial K} - \mathbb{A}_3^K (\mathbb{A}_1^K)^{-1} \mathbb{A}_f^K + \tau \mathrm{DD}^K \mathbf{u}_{\partial K} \\
&= \mathbb{C}^K \mathbf{u}_{\partial K} - \mathbb{C}_f^K,
\end{aligned}
\tag{5.7}
$$

where $\mathbf{u}_{\partial K}$ is the vector of degrees of freedom of $\widehat{u}_h|_{\partial K}$.

Note on implementation. Construction of the local solvers (5.5) and (5.6), as well as recovery of internal values using (5.3), requires looping over elements. However, this can be easily done in parallel, since at this stage there is no interconnection between elements. Note that we have avoided looping over elements in all previous computations, requiring frequent access to coefficients and geometric features.

6 Boundary conditions and global solver

Dirichlet boundary conditions. The discrete Dirichlet boundary conditions require finding the decompositions

$$\widehat{u}_h|_e = \sum_{j=1}^{d_2} u_j^e D_j^e, \qquad e \in \mathcal{E}_h^D$$

by solving the system

$$\sum_j \left(\int_e D_i^e D_j^e \right) u_j^e = \int_e D_i^e u_D, \qquad i = 1, \ldots, d_2, \qquad e \in \mathcal{E}_h^D.$$

Using a quadrature rule on the reference element, and parametrizing from it, we have to solve the approximate system

$$|e| \sum_{j=1}^{d_2} \Big(\sum_{r=1}^{N_{\mathrm{qd2}}} \varpi_r \widehat{D}_i(\widehat{\mathbf{q}}_r) \widehat{D}_j(\widehat{\mathbf{q}}_j) \Big) u_j^e = |e| \sum_{r=1}^{N_{\mathrm{qd2}}} \varpi_r \widehat{D}_i(\widehat{\mathbf{q}}_r) u_D(\mathbf{q}_r^e). \tag{6.1}$$

Evaluation of the data function u_D at all the quadrature points is done with a similar strategy to the one used for source terms (3.4). We start by organizing nodal information in three $3 \times N_{\mathrm{dir}}$ (recall that $N_{\mathrm{dir}} = \#\mathcal{E}_h^D$) matrices X^D, Y^D, Z^D, each of the containing the

corresponding coordinates of the nodes of each of the Dirichlet faces. If Ξ is the $N_{\mathrm{qd2}} \times 3$ matrix with the barycentric coordinates of the quadrature points in $\widehat{K}_2$ (Section 4), then the $N_{\mathrm{qd2}} \times N_{\mathrm{dir}}$ matrices

$$\mathrm{X}^{\mathrm{dir}} := \Xi\, \mathrm{X}^D, \qquad \mathrm{Y}^{\mathrm{dir}} := \Xi\, \mathrm{Y}^D, \qquad \mathrm{Z}^{\mathrm{dir}} := \Xi\, \mathrm{Z}^D, \tag{6.2}$$

contain the coordinates of the quadrature points on the Dirichlet faces. Using the matrix in (4.7) (see also (4.8)), it is clear that (6.1) can be implemented by solving a system with multiple right-hand sides

$$\widehat{\mathbf{u}}_D := \big((\boldsymbol{\varpi} \odot \mathrm{D})^\top \mathrm{D}\big)^{-1} (\boldsymbol{\varpi} \odot \mathrm{D})^\top u_D(\mathrm{X}^{\mathrm{dir}}, \mathrm{Y}^{\mathrm{dir}}, \mathrm{Z}^{\mathrm{dir}}). \tag{6.3}$$

The result is a $d_2 \times N_{\mathrm{dir}}$ matrix.

Neumann boundary conditions. As opposed to Dirichlet conditions (that are essential in this formulation), Neumann boundary conditions will appear in the right-hand side of the global system. Our goal is to compute the integrals (recall that $|\mathbf{n}_e| = |e|$)

$$\int_e (\boldsymbol{g}_N \cdot \boldsymbol{\nu}_e) D_i^e \approx \sum_{r=1}^{N_{\mathrm{qd2}}} \varpi_r \widehat{D}_i(\widehat{\mathbf{q}}_r)\, \boldsymbol{g}_N(\mathbf{q}_r^e) \cdot \mathbf{n}_e \qquad i = 1, \ldots, d_2, \qquad e \in \mathcal{E}_h^N.$$

If we consider matrices $\mathrm{X}^{\mathrm{neu}}$, $\mathrm{Y}^{\mathrm{neu}}$, $\mathrm{Z}^{\mathrm{neu}}$, defined as in (6.2) (but using nodal information for Neumann faces), and if $\mathbf{n}_x$, $\mathbf{n}_y$, $\mathbf{n}_z$ are $N_{\mathrm{neu}} \times 1$ (recall that $N_{\mathrm{neu}} = \#\mathcal{E}_h^N$) column vectors with the components of the vectors $\mathbf{n}_e$ for $e \in \mathcal{E}_h^N$, then everything is done with the simple computation

$$\boldsymbol{\Phi}_N := \sum_{\star \in \{x,y,z\}} \mathbf{n}_\star^\top \odot \big((\boldsymbol{\varpi} \odot \mathrm{D})^\top g_\star(\mathrm{X}^{\mathrm{neu}}, \mathrm{Y}^{\mathrm{neu}}, \mathrm{Z}^{\mathrm{neu}})\big), \tag{6.4}$$

where $\boldsymbol{g}_N = (g_x, g_y, g_z)$. Note that if the Neumann boundary condition is given in a more standard way $-\mathbf{q} \cdot \boldsymbol{\nu} = g_N$, then the computation is slightly simpler

$$\boldsymbol{\Phi}_N := \mathbf{area}_N^\top \odot \big((\boldsymbol{\varpi} \odot \mathrm{D})^\top g_N(\mathrm{X}^{\mathrm{neu}}, \mathrm{Y}^{\mathrm{neu}}, \mathrm{Z}^{\mathrm{neu}})\big)$$

where $\mathbf{area}_N$ contains the areas of all Neumann faces.

Assembly process. The local solvers produce a $4d_2 \times 4d_2 \times N_{\mathrm{elt}}$ array $\mathbb{C}$. We now use the `sparse` MATLAB builder to assembly the global matrix. The degrees of freedom associated to face $e \in \{1, \ldots, N_{\mathrm{fc}}\}$ are

$$\mathrm{list}(e) := (e-1)d_2 + \{1, \ldots, d_2\}.$$

The degrees of freedom associated to the faces of K are thus

$$\mathrm{dof}(K) := \{\mathrm{list}(e_1^K), \mathrm{list}(e_2^K), \mathrm{list}(e_3^K), \mathrm{list}(e_4^K)\}.$$

We then create two new $4d_2 \times 4d_2 \times N_{\mathrm{elt}}$ arrays

$$\mathrm{Row}_{ij}^K = \mathrm{dof}(K)_i \qquad \mathrm{Col}_{ij}^K = \mathrm{dof}(K)_j, \quad \text{so that} \quad (\mathrm{Col}^K)^\top = \mathrm{Row}^K.$$

The (i, j) element of $\mathbb{C}^K$ has to be assembled at the location $(\text{Row}_{ij}^K, \text{Col}_{ij}^K) = (\text{dof}(K)_i, \text{dof}(K)_j)$. The result is a sparse $d_2 N_{\text{fc}} \times d_2 N_{\text{fc}}$ matrix $\mathbb{H}$. This matrix collects the fluxes (5.7) for all the elements, with the result that opposing sign fluxes in internal faces (the normal vector points in different directions) are added. The assembly of the source term, given in the matrix $\mathbb{C}_f$, can be carried out using the `accumarray` command. The element $(\mathbb{C}_f^K)_i$ has to be added to the location $\text{dof}(K)_i$. The result is a vector $\mathbf{F}$ with $d_2 N_{\text{fc}}$ components. Let us consider the system at its current stage

$$\mathbb{H}\,\widehat{\mathbf{u}} = \mathbf{F} + \mathbf{G}_N, \tag{6.5}$$

where $\mathbf{G}_N$ is the $d_2 N_{\text{fc}}$ vector containing the elements of $\mathbf{\Phi}_N$ in the degrees of freedom corresponding to Neumann faces and zeros everywhere else. This is the matrix representation of the system (1.2) with no Dirichlet boundary conditions, i.e., assuming homogeneous Neumann boundary conditions on Γ_N, the system having been written in the $\widehat{u}_h$ variable after local inversion of (1.2a)-(1.2b).

What is left is the standard elimination of Dirichlet degrees of freedom from (6.5), namely values of Dirichlet faces are taken from (6.3) and sent to the right-hand side of the system, and rows corresponding to Dirichlet degrees of freedom are ignored.

Reconstruction. The solution of the resulting system is $\widehat{u}_h \in M_h$. Reconstruction of the other variables $(\boldsymbol{q}_h, u_h)$ is done by solving local problems. In matrix form, we have to solve on each $K \in \mathcal{T}_h$ the system

$$\mathbb{A}_1^K \begin{bmatrix} \mathbf{q}_K \\ \mathbf{u}_K \end{bmatrix} = \mathbb{A}_f^K - \mathbb{A}_2^K \mathbf{u}_{\partial K}.$$

This can be done in parallel.

7 Add-ons

Matrices for convection-diffusion problems. With very similar techniques, it is easy to compute convection matrices with variable coefficients

$$\int_K m\, P_i^K \partial_\star P_j^K, \qquad i, j = 1, \ldots, d_3, \qquad K \in \mathcal{T}_h, \tag{7.1}$$

as well as surface matrices with variable coefficients

$$\frac{\xi_\ell^K}{|e_\ell^K|} \int_{e_\ell^K} \alpha\, D_i^{e_\ell^K} P_j^K, \qquad \begin{aligned} i &= 1, \ldots, d_2, \\ j &= 1, \ldots, d_3, \end{aligned} \quad \ell \in \{1, 2, 3, 4\}, \quad K \in \mathcal{T}_h,$$

$$\frac{\xi_\ell^K}{|e_\ell^K|} \int_{e_\ell^K} \alpha\, D_i^{e_\ell^K} D_j^{e_\ell^K}, \qquad i, j, = 1, \ldots, d_2, \quad \ell \in \{1, 2, 3, 4\}, \quad K \in \mathcal{T}_h.$$

These matrices are needed for coding HDG applied to convection-diffusion problems [3], which needs the bilinear forms

$$(\boldsymbol{\beta} \cdot \nabla u_h, w_h)_K, \quad \langle (\boldsymbol{\beta} \cdot \boldsymbol{\nu}) u_h, \widehat{v}_h \rangle_{\partial K}, \quad \text{and} \quad \langle (\boldsymbol{\beta} \cdot \boldsymbol{\nu}) \widehat{u}_h, \widehat{v}_h \rangle_{\partial K}.$$

Postprocessing. If we look for $u_h^* : \Omega \to \mathbb{R}$ such that $u_h^*|_K \in \mathcal{P}_{k+1}(K)$ and for all $K \in \mathcal{T}_h$,

$$(\nabla u_h^*, \nabla w_h)_K = -(\kappa^{-1} \boldsymbol{q}_h, \nabla w_h)_K \qquad \forall w_h \in \mathcal{P}_{k+1}(K), \tag{7.2a}$$

$$(u_h^*, 1)_K = (u_h, 1)_K, \tag{7.2b}$$

then it can be shown that this local postprocessed approximation has one additional order of convergence [5]. In order to compute this postprocessing we have to use matrices of the form (7.1) in the right-hand side (using an additional polynomial degree) and we need to compute local stiffness matrices

$$\int_K \nabla P_i^K \cdot \nabla P_j^K \qquad i, j, = 1, \ldots, d_3(k+1), \qquad K \in \mathcal{T}_h.$$

As in the computation of the convection matrices (3.11), this can be done using geometric vectors and Kronecker products.

Local L^2 projections. For several different purposes, it is also convenient to have some local projections at hand. The first one is the $L^2(\Omega)$ projection on W_h: given f we compute $f_h \in W_h$ such that

$$(f_h, w_h)_K = (f, w_h)_K \qquad \forall w_h \in \mathcal{P}_k(K) \qquad \forall K \in \mathcal{T}_h.$$

Using (3.4) and (3.5), and up to quadrature errors, this projection is easily computed with a single instruction

$$\big((\widehat{\boldsymbol{\omega}} \odot \mathrm{P})^T \mathrm{P}\big)^{-1} (\widehat{\boldsymbol{\omega}} \odot \mathrm{P})^\top f(\mathrm{X}, \mathrm{Y}, \mathrm{Z}).$$

The L^2 projection on M_h

$$\langle f_h, \widehat{v}_h \rangle_e = \langle f, \widehat{v}_h \rangle_e \qquad \forall \widehat{v}_h \in \mathcal{P}_k(e) \qquad \forall e \in \mathcal{E}_h, \tag{7.3}$$

is computed using a formula like (6.3)

$$\big((\boldsymbol{\varpi} \odot \mathrm{D})^\top \mathrm{D}\big)^{-1} (\boldsymbol{\varpi} \odot \mathrm{D})^\top u_D(\mathrm{X}^{\mathrm{all}}, \mathrm{Y}^{\mathrm{all}}, \mathrm{Z}^{\mathrm{all}}),$$

where we use quadrature points on all faces of the triangulation (see (6.2) for the Dirichlet case).

Error functions. Once again with very similar ideas it is easy to code the computation of errors

$$\int_\Omega |u - u_h|^2 \qquad \sum_{e \in \mathcal{E}_h} |e| \int_e |\widehat{u}_h - u|^2,$$

for a given function u and approximations $u_h \in W_h$ and $\widehat{u}_h \in M_h$.

HDG projection. A final projection is directly tied to the HDG method. The input is the collection $(\boldsymbol{q}, u)$ of a vector field an a scalar function. The output are functions $(\boldsymbol{q}_h, u_h) \in \boldsymbol{V}_h \times W_h$ satisfying

$$(\boldsymbol{q}_h, \boldsymbol{r})_K = (\boldsymbol{q}, \boldsymbol{r})_K \qquad \forall \boldsymbol{r} \in \mathcal{P}_{k-1}(K)^3 \qquad \forall K \in \mathcal{T}_h, \qquad (7.4a)$$

$$(u_h, w_h)_K = (u, w_h)_K \qquad \forall w_h \in \mathcal{P}_{k-1}(K) \qquad \forall K \in \mathcal{T}_h, \qquad (7.4b)$$

$$\langle \boldsymbol{q}_h \cdot \boldsymbol{\nu} + \tau_K u_h, \widehat{v}_h \rangle_{\partial K} = \langle \boldsymbol{q} \cdot \boldsymbol{\nu} + \tau u, \widehat{v}_h \rangle_{\partial K} \qquad \forall \widehat{v}_h \in \mathcal{R}_k(\partial K) \qquad \forall K \in \mathcal{T}_h. \qquad (7.4c)$$

It has to be understood that the first two groups of equations are void when $k = 0$. If we construct a mass matrix (with constant unit mass) M^K and drop the last $d_2 = \dim \mathcal{P}_k(K) - \dim \mathcal{P}_{k-1}(K)$ rows (recall that local bases are hierarchical), we obtain a $(d_3 - d_2) \times d_3 \times N_{\mathrm{elt}}$ matrix with slices $\widetilde{\mathrm{M}}^K$

$$\int_K P_i^K P_j^K, \qquad i = 1, \dots, d_3 - d_2, \qquad j = 1, \dots, d_3.$$

Using the surface matrices of Section 5, we are led to solve local linear systems with matrices:

$$\begin{bmatrix} \widetilde{\mathrm{M}}^K & \mathrm{O} & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \widetilde{\mathrm{M}}^K & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \widetilde{\mathrm{M}}^K & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \mathrm{O} & \widetilde{\mathrm{M}}^K \\ n_x \mathrm{DP}^K & n_y \mathrm{DP}^K & n_z \mathrm{DP}^K & \tau \mathrm{DP}^K \end{bmatrix}.$$

The corresponding right-hand sides can be easily constructed using the techniques of previous sections.

BDM. A hybridized coding of the three dimensional Brezzi-Douglas-Marini element (more properly speaking, this is an element by Brezzi-Douglas-Durán-Fortin, discovered simultaneously by Nédélec) is also easily attainable. For this case, we take $k \geqslant 1$, define

$$W_h := \prod_{K \in \mathcal{T}_h} \mathcal{P}_{k-1}(K), \qquad \boldsymbol{V}_h := \prod_{K \in \mathcal{T}_h} \mathcal{P}_k(K)^3$$

and keep M_h as before. The mixed BDM approximation to (1.1) uses equations (1.2) with two simple modifications: $\tau \equiv 0$, and equation (1.2a) is only tested in $\mathcal{P}_{k-1}(K)^3$. At the implementation level, this means that we only need to redefine the local solvers. Since $\dim \mathcal{P}_{k-1}(K) = d_3 - d_2$ and the only unknown that is in a smaller space is u_h, we only need to eliminate: the last d_2 rows and columns of $\mathbb{A}_1^K$, the last d_2 rows of $\mathbb{A}_2^K$ and $\mathbb{A}_f^K$, and the last d_2 columns of $\mathbb{A}_3^K$. (This can be done by erasing the corresponding parts of the three dimensional arrays where we have stored the HDG matrices.) All other parts of the HDG code remain untouched.

8 Experiments

We next give some convergence tests for the method. We take Ω to be the polyhedron sketched in Figure 1. The faces of the polyhedron corresponding to $z = 0$, $z = 1$ and
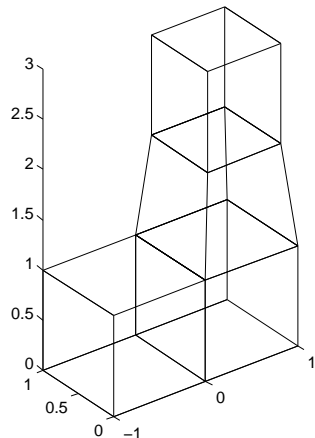
N_{elt}	24	192	1536	12288
N_{fc}	66	456	3360	25728

Figure 1: Domain for the experiments and data for the four tetrahedrizations used in the experiments.

$z = 3$ conform the Dirichlet boundary Γ_D. The coarsest triangulation –obtained by a tetrahedral partition of each the four hexahedra shown in Figure 1–, contains 24 elements. Three nested refinements of this partition are used. The main triangulation data are then given in Figure 1. We use variable coefficients:

$$\kappa = 2 + \sin x \sin y \sin z, \qquad c = 1 + \tfrac{1}{2}(x^2 + y^2 + z^2),$$

and take data so that $u = \sin(x\,y\,z)$ is the exact solution.

We test for several values of k on the four triangulations. Tables 1–3 show relative errors

$$e_h^q := \frac{\|\boldsymbol{q} - \boldsymbol{q}_h\|_\Omega}{\|\boldsymbol{q}\|_\Omega} \qquad e_h^u := \frac{\|u - u_h\|_\Omega}{\|u\|_\Omega}, \qquad e_h^{\hat{u}} := \frac{\|u - \widehat{u}_h\|_h}{\|u\|_h} \tag{8.1}$$

and relative errors for superconvergent quantities

$$\varepsilon_h^u := \frac{\|\Pi u - u_h\|_\Omega}{\|u\|_\Omega}, \qquad \varepsilon_h^{\hat{u}} := \frac{\|P u - \widehat{u}_h\|_h}{\|u\|_h}, \qquad e_h^\star := \frac{\|u - u_h^\star\|_\Omega}{\|u\|_\Omega}. \tag{8.2}$$

Here: $\|u\|_h^2 := \sum_{e \in \mathcal{E}_h} |e| \, \|u\|_e^2$, $u_h^\star$ is the postprocessed solution defined by (7.2), $Pu \in M_h$ is the $L^2(\partial \mathcal{T}_h)$ projection defined in (7.3) and $\Pi u \in W_h$ is the scalar component of the projection $(\Pi \boldsymbol{q}, \Pi u)$ defined in (7.4). Theory [5] shows that for smooth solutions, the errors (8.1) behave like $\mathcal{O}(h^{k+1})$ while errors (8.2) behave like $\mathcal{O}(h^{k+2})$ except when $k = 0$, where they behave like $\mathcal{O}(h)$. Estimates of order of convergence for a general quantity e_h are computed using the formula $\log_2 e_{h/2}/\log_2 e_h$.

We finally test the validity of the HDG method as a p-method, by fixing the tetrahedrization (the second one in Figure 1) and increasing k from 0 to 3. We compute the relative errors (8.1) as functions of k and check whether the rates

$$\frac{\log(e_k/e_{k+1})}{\log(e_{k+1}/e_{k+2})} \approx 1, \tag{8.3}$$

20

e_h^q	e.c.r.	e_h^u	e.c.r.	$e_h^{\hat{u}}$	e.c.r.
6.9694e-001	–	1.2404e+000	–	5.7265e-001	–
4.2490e-001	0.71	8.5011e-001	0.55	3.2710e-001	0.81
2.2749e-001	0.90	5.0782e-001	0.74	1.7587e-001	0.90
1.1739e-001	0.95	2.8336e-001	0.85	9.1779e-002	0.94

ε_h^u	e.c.r.	$\varepsilon_h^{\hat{u}}$	e.c.r.	$\varepsilon_h^{\star}$	e.c.r.
1.3127e+000	–	5.4851e-001	–	1.2515e+000	–
6.4689e-001	1.02	2.9624e-001	0.89	8.4019e-001	0.57
2.6697e-001	1.28	1.5688e-001	0.92	5.0151e-001	0.74
1.1100e-001	1.27	8.1482e-002	0.95	2.8027e-001	0.84

Table 1: Errors for different triangulations (see Figure 1) with the lowest order method $k = 0$.

e_h^q	e.c.r.	e_h^u	e.c.r.	$e_h^{\hat{u}}$	e.c.r.
1.3607e-001	–	4.2677e-001	–	1.3580e-001	–
3.6794e-002	1.89	1.2953e-001	1.72	3.1932e-002	2.09
9.6645e-003	1.93	3.6956e-002	1.81	7.9337e-003	2.01
2.4878e-003	1.96	9.9288e-003	1.90	1.9876e-003	2.00

ε_h^u	e.c.r.	$\varepsilon_h^{\hat{u}}$	e.c.r.	$\varepsilon_h^{\star}$	e.c.r.
2.9210e-001	–	4.6150e-002	–	3.3850e-002	–
4.5051e-002	2.70	5.8813e-003	2.97	4.4054e-003	2.94
6.5777e-003	2.78	7.6131e-004	2.95	5.5048e-004	3.00
8.9450e-004	2.88	9.6512e-005	2.98	6.8066e-005	3.02

Table 2: Errors for different triangulations (see Figure 1) with the lowest order method $k = 1$. All quantities in (8.2) are shown to be superconvergent.

as would be expected. Note that the theory for p-convergence of HDG is not fully developed. The results are reported in Table 4.

Other easy benchmarks for the method are exact polynomial solutions. These have been tried on the implementation as a way to test exactness of approximation and quadrature in the process.

References

[1] D. N. Arnold and F. Brezzi. Mixed and nonconforming finite element methods: implementation, postprocessing and error estimates. *RAIRO Modél. Math. Anal. Numér.*, 19(1):7–32, 1985.

[2] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York, 1991.

e_h^q	e.c.r.	e_h^u	e.c.r.	$e_h^{\hat{u}}$	e.c.r.
2.7400e-002	–	6.5553e-002	–	1.9182e-002	–
4.0693e-003	2.75	1.1591e-002	2.50	2.4377e-003	2.98
5.4030e-004	2.91	1.6402e-003	2.82	3.0858e-004	2.98
6.8953e-005	2.97	2.1698e-004	2.92	3.8901e-005	2.99
ε_h^u	e.c.r.	$\varepsilon_h^{\hat{u}}$	e.c.r.	$\varepsilon_h^{\star}$	e.c.r.
2.7092e-002	–	4.7030e-003	–	4.7177e-003	–
3.1232e-003	3.12	3.5134e-004	3.74	3.5580e-004	3.73
2.1668e-004	3.85	2.2499e-005	3.96	2.2835e-005	3.96
1.4237e-005	3.93	1.3859e-006	4.02	21.4212e-006	4.01

Table 3: Errors for different triangulations (see Figure 1) with the lowest order method $k = 2$. All quantities in (8.2) are shown to be superconvergent.

k	$e_h^q(k)$	e.c.r.	$e_h^u(k)$	e.c.r.	$e_h^{\hat{u}}(k)$	e.c.r.
0	4.2490e-001	–	8.5011e-001	–	3.2710e-001	–
1	3.6794e-002	–	1.2953e-001	–	3.1932e-002	–
2	4.0693e-003	1.11	1.1591e-002	0.78	2.4377e-003	0.90
3	4.4704e-004	1.00	1.3590e-003	1.13	1.7465e-004	0.98

Table 4: History of convergence for increasing polynomial degrees, and convergence test following (8.2).

[3] Y. Chen and B. Cockburn. Analysis of variable-degree HDG methods for convection-diffusion equations. Part I: general nonconforming meshes. *IMA J. Numer. Anal.*, 32(4):1267–1293, 2012.

[4] B. Cockburn, J. Gopalakrishnan, and R. Lazarov. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM J. Numer. Anal.*, 47(2):1319–1365, 2009.

[5] B. Cockburn, J. Gopalakrishnan, and F.-J. Sayas. A projection-based error analysis of HDG methods. *Math. Comp.*, 79(271):1351–1367, 2010.

[6] V. J. Ervin. Computational bases for RT_k and BDM_k on triangles. *Comput. Math. Appl.*, 64(8):2765–2774, 2012.

[7] C. A. Felippa. A compendium of FEM integration formulas for symbolic work. *Engineering Computations*, 21(8):867–890, 2004.

[8] R. A. Kirby. Singularity-free evaluation of collapsed-coordinate orthogonal polynomials. *ACM Trans. Math. Softw.*, 37(1):Article No. 5, 2010.

[9] R. M. Kirby, S. J. Sherwin, and B. Cockburn. To CG or to HDG: a comparative study. *J. Sci. Comput.*, 51(1):183–212, 2012.

[10] R. Stenberg. Postprocessing schemes for some mixed finite elements. *RAIRO Modél. Math. Anal. Numér.*, 25(1):151–167, 1991.

[11] L. Zhang, T. Cui, and H. Liu. A set of symmetric quadrature rules on triangles and tetrahedra. *J. Comput. Math.*, 27(1):89–96, 2009.