A Parallel and Adaptive Hybridized Discontinuous Galerkin Method for Anisotropic Nonhomogeneous Diffusion

Ali Samii[a,b,*], Craig Michoski[a], Clint Dawson[a,b]

[a] *Computational Hydraulics Group (CHG), Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, Austin, TX 78712.*
[b] *Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, TX 78712.*

Abstract

The diffusion equation in anisotropic and nonhomogeneous media arises in the study of flow through porous media with sharp material interfaces. We discuss the solution of this problem by a hybrid discontinuous Galerkin (HDG) method. The method can be applied in three steps. First, we use a condensation technique to derive the scalar variable and the flux inside each element in terms of the numerical trace on the faces of that element. Then we form a global system of equations to solve for these numerical traces. We then solve the equation inside each element for the internal unknowns using the obtained numerical traces in the global solve step. Similar to other DG variants, HDG is a locally conservative method, and a noticeable share of calculations are performed independently within each element. In a mesh with pth order elements ($p \geq 0$), this method gives $p+1$ order of accuracy for smooth solutions for both the scalar variable and the flux. Moreover, by using a simple post-processing technique, one can reach an accuracy of order $p+2$ for the scalar variable. To be able to handle problems with sharp material discontinuities, we use an adaptive refinement strategy, with octree grid structure. Hence, we avoid a larger global system which would arise from a fine uniform grid. In this process, we refine those elements with highest gradient of flux, at two sides of their faces. We have also utilized shared and distributed memory parallelism to enhance the performance of the method. The method is implemented using different modules of deal.II [1], PETSc [2], p4est [3] and Hypre [4]. To demonstrate the accuracy, efficiency, scalability, and flexibility of the method, several two and three dimensional numerical experiments are studied.

Keywords: Finite element method, Hybrid methods, Strong nonhomogeneity, Anisotropy, Porous media.

1. Introduction

In this paper, we study the hybrid discontinuous Galerkin (HDG) method [5] for diffusion problems arising in single phase flow through heterogeneous porous media. The HDG method, as the name implies, is a hybrid form of the discontinuous Galerkin (DG) method, whereby the solution within each element is expressed in terms of trace unknowns that are defined on the element faces. The trace unknowns are determined by solving a global, symmetric and positive definite linear system of equations. Given the trace variables the solution variables inside each element can be found independently. In this sense, the HDG method is very similar to the hybridized form of the mixed finite element method [6], without the compatibility constraints on the velocity and pressure spaces imposed in the traditional mixed method. See [5] for a thorough discussion on the relationship between these two methods.

The HDG improves upon earlier DG methods, such as the local discontinuous Galerkin (LDG) [7] and Interior Penalty (IP) [8] methods, in several respects. The method is convergent for lowest order $p = 0$

*Corresponding author
Email addresses: samii@ices.utexas.edu (Ali Samii), michoski@ices.utexas.edu (Craig Michoski), clint@ices.utexas.edu (Clint Dawson)

spaces, whereas earlier DG methods required $p \geq 1$ (except for one rather impractical variant of the LDG method where one must solve for both velocity and pressure unknowns simultaneously, see [7, 9]). This property is especially relevant to porous media problems, where many simulators utilize lowest order spaces. Furthermore, the method is of optimal order $(p + 1)$ for both the velocity and pressure unknowns, and compared to earlier DG methods, the size of the global system of equations to be solved is reduced, due to solving for trace unknowns rather than element unknowns. Similar to the LDG and symmetric IP methods, given a symmetric elliptic operator, the HDG gives rise to a symmetric matrix. This is not the case for other IP methods, such as the nonsymmetric and incomplete interior penalty methods [8, 10]. The HDG method also retains all of the favorable aspects of DG methods, in the sense that it is locally conservative, which is crucial for geoscience applications, it allows for very general unstructured meshes including nonconforming meshes, and is highly parallelizable due to the local nature of much of its computation. In fact, as we shall show, as the polymomial order of the method increases, much of the work of the method is done within each element.

The HDG method is also comparable to other locally conservative, unstructured mesh methods which have seen increased interest in recent years, including the mimetic finite difference (MFD) methods and the multipoint flux approximation (MPFA) methods.

MFD methods have been used to solve the diffusion equation in strongly nonhomogeneous and anisotropic media [11, 12]. The main idea in this class of methods is to construct a set of discrete difference operators, which can mimic the main properties of the original continuum differential operators, e.g. local conservation, solution symmetry, null space consistency and the fundamental theorems in vector and tensor calculus. Moreover, the discrete operators preserve the symmetry and ellipticity of the diffusion operator by producing a symmetric positive definite global matrix. In [13], it was shown that the mimetic discretization is second order accurate for the scalar variable and the flux, under suitable regularity assumptions of the solution and the grid geometry. This study was later extended to polyhedral meshes with curved faces [14]. In [15], a mimetic discretization was introduced for non-convex and degenerate polygonal grids. In [16], instead of a constant distribution, a linear variation was considered for the flux on the faces of cells, which results in a higher order approximation for both scalar variable and the flow. Generally, the mimetic methods have great flexibility in the design of the mesh geometry. However, their literature lacks extension to arbitrary higher order approximations.

MPFA methods (see, for example [17–23]) are extensions of the traditional block-centered finite difference method (or mixed finite element methods with special quadrature), commonly used in petroleum reservoir simulators, to unstructured meshes. They share many similarities to mixed finite element and MFD methods in that velocity unknowns are defined on edges and expressed in terms of pressure unknowns defined within each cell. MPFA methods are also locally conservative and have now been extended to complex two-dimensional and three-dimensional geometries. As with MFD methods, they are generally restricted to low order approximations.

We remark that extensions and improvements of MPFA and MFD methods are very active areas of research and the literature on these methods continues to grow rapidly. We have not by any means provided an exhaustive discussion of these approaches.

This paper is organized as follows. In the second section, we introduce the governing equation and the corresponding boundary conditions in our problem. The notation and the approximation spaces for solution and test functions are also presented here. In section 3 we explain our implementation and the three major stages of hybrid DG to solve the problem. We also review some of the properties of the local and global matrices (such as positive definiteness) which can be used to construct a more efficient solver. The local post-processing technique which is used to enhance the convergence rate of the scalar variable is presented in section 4. In section 5, we have included a brief review of the octree grid structure and how we have employed it in our implementation. Especially, we explain our technique to make the skeleton space on the octrees conforming with the space of single-valued trace. Section 6 covers the available parallelism opportunities in the method, and how one can mix shared and distributed memory parallelism to achieve a more efficient implementation. Finally, in section 7, we present an extensive set of two and three dimensional numerical experiments to study the accuracy, efficiency, and scalability of the described method.

2. Problem Statement and Space Discretization

Consider the steady state diffusion equation in the domain $\Omega \subset \mathbb{R}^d$:

$$-\nabla \cdot (\boldsymbol{\kappa} \nabla u) = f \qquad \text{in } \Omega, \tag{1}$$

subject to the following boundary conditions:

$$\begin{aligned} u &= g_D \qquad \text{on } \partial\Omega_D, \\ (-\boldsymbol{\kappa}\nabla u) \cdot \mathbf{n} &= g_N \qquad \text{on } \partial\Omega_N. \end{aligned} \tag{2}$$

Here, $\partial\Omega$ denotes the boundary of the domain, which is comprised of disjoint sets $\partial\Omega_D$ and $\partial\Omega_N$. $\boldsymbol{\kappa} = \boldsymbol{\kappa}(\mathbf{x})$ is the symmetric positive definite tensor of diffusivity coefficients (κ_{ij}), and $f = f(\mathbf{x})$ is the source term. When modeling the flow in a porous medium, u is the hydraulic head, and $\boldsymbol{\kappa}$ is the permeability tensor divided by the fluid viscosity. Generally, $\boldsymbol{\kappa}$ can have a discontinuous variation in space, and f depends on the distribution of the sources and sinks in the domains. However, in order to make the integrals in the variational formulation of this problem meaningful, we assume $\kappa_{ij} \in L^\infty(\Omega)$, and $f \in L^2(\Omega)$.

Next, we introduce $\mathbf{q} = -\boldsymbol{\kappa}\nabla u$, and form the system of first-order equations corresponding to (1):

$$\begin{aligned} \mathbf{q} + \boldsymbol{\kappa}\nabla u &= 0 \\ \nabla \cdot \mathbf{q} &= f \end{aligned} \qquad \text{in } \Omega, \tag{3}$$

with the boundary conditions:

$$\begin{aligned} u &= g_D \qquad \text{on } \partial\Omega_D, \\ \mathbf{q} \cdot \mathbf{n} &= g_N \qquad \text{on } \partial\Omega_N. \end{aligned} \tag{4}$$

2.1. Notation

Let $\mathcal{T}_h = \{K\}$ be a finite collection of disjoint elements partitioning Ω. We use quadrilateral elements for $d = 2$, and hexahedral elements for $d = 3$. Let $\partial\mathcal{T}_h := \{\partial K : K \in \mathcal{T}_h\}$, and use $\mathcal{E}_h^0$ to denote the set of interior faces, and $\mathcal{E}_h^\partial$ for the set of boundary faces. Moreover $\mathcal{E}_h = \mathcal{E}_h^0 \cup \mathcal{E}_h^\partial$. On a given interior face $e \in \mathcal{E}_h^0$, we will assign K^+ and K^- as the elements on the two sides of e, i.e. $e = \partial K^+ \cap \partial K^-$; the values of $(u, \mathbf{q})$ on the common face of these elements will be denoted by $(u^\pm, \mathbf{q}^\pm)$. Similarly, $\mathbf{n}^\pm$ are the outward unit normals corresponding to $K^\pm$.

The mean and jump of the scalar-valued (u) and vector-valued information $(\mathbf{q})$ at $e \in \mathcal{E}_h^0$ are defined as:

$$\begin{aligned} \{\!\{u\}\!\} &= (u^+ + u^-)/2\,, \qquad & [\![u\mathbf{n}]\!] &= u^+\mathbf{n}^+ + u^-\mathbf{n}^-, \\ \{\!\{\mathbf{q}\}\!\} &= (\mathbf{q}^+ + \mathbf{q}^-)/2\,, \qquad & [\![\mathbf{q}\cdot\mathbf{n}]\!] &= \mathbf{q}^+\cdot\mathbf{n}^+ + \mathbf{q}^-\cdot\mathbf{n}^-. \end{aligned}$$

For boundary faces where $(u, \mathbf{q})$ and $\mathbf{n}$ is single-valued, the mean and jumps are defined as:

$$\begin{aligned} \{\!\{u\}\!\} &= u\,, \qquad & [\![u\mathbf{n}]\!] &= u\mathbf{n}, \\ \{\!\{\mathbf{q}\}\!\} &= \mathbf{q}\,, \qquad & [\![\mathbf{q}\cdot\mathbf{n}]\!] &= \mathbf{q}\mathbf{n}. \end{aligned}$$

2.2. Approximation Spaces

For each element $K \in \mathcal{T}_h$ and $p \geq 0$, Let $\mathcal{P}^p(K)$ denote the space of polynomials of degree at most p in each direction. For vector valued functions we define $\boldsymbol{\mathcal{P}}^p(G) := (\mathcal{P}^p(G))^d$. The approximation spaces in Ω are chosen as the set of square integrable functions in Ω, with their restriction to the domain of each element (K) belong to $\mathcal{P}^p(K)$ or $\boldsymbol{\mathcal{P}}^p(K)$; i.e.

$$W_h^p := \{w \in L^2(\Omega) : w|_K \in \mathcal{P}^p(K) \quad \forall K \in \mathcal{T}_h\}, \tag{5a}$$

$$\mathbf{V}_h^p := \{\mathbf{v} \in (L^2(\Omega))^d : \mathbf{v}|_K \in \boldsymbol{\mathcal{P}}^p(K) \quad \forall K \in \mathcal{T}_h\}. \tag{5b}$$

The approximation space on the mesh skeleton ($\mathcal{E}_h$) is also defined in a similar fashion:

$$M_h^p := \{\mu \in L^2(\mathcal{E}_h) : \mu|_e \in \mathcal{P}^p(e) \quad \forall e \in \mathcal{E}_h\}. \tag{5c}$$

We will also use another space on the skeleton, with a built-in boundary condition on $\partial\Omega_D$:

$$M_h^p(g) := \{\mu \in L^2(\mathcal{E}_h) : \mu|_e \in \mathcal{P}^p(e) \quad \forall e \in \mathcal{E}_h,\ \mu|_{\partial\Omega_D} = \Pi_\partial g\}. \tag{5d}$$

Here Π_∂ denotes an L^2-projection acting on a given $\zeta \in L^2(\mathcal{E}_h)$, and the restriction of $\Pi_\partial\zeta$ to $e \in \mathcal{E}_h$ is in $\mathcal{P}^p(e)$, and satisfies:

$$\langle \Pi_\partial\zeta - \zeta, \mu\rangle_e = 0, \qquad \forall \mu \in \mathcal{P}^p(e).$$

We will denote the inner product of functions v and w in $G \subset \mathbb{R}^d$ by $(v,w)_G$, i.e. $(v,w)_G = \int_G vw \, \mathrm{d}G$. Moreover, $\langle v, w\rangle_\Gamma$ is used to denote $\int_\Gamma vw \, \mathrm{d}\Gamma$, when $\Gamma \subset \mathbb{R}^{d-1}$.

3. Solution Approach

We seek a piecewise polynomial solution $(u_h, \mathbf{q}_h) \in W_h^p \times \mathbf{V}_h^p$ to the variational formulation corresponding to equation (3). Hence, for all $(w, \mathbf{v}) \in W_h^p \times \mathbf{V}_h^p$ we want our solution to satisfy:

$$\begin{aligned} (\boldsymbol{\kappa}^{-1}\mathbf{q}_h, \mathbf{v})_K - (u_h, \nabla\cdot\mathbf{v})_K + \langle \hat{u}_h, \mathbf{v}\cdot\mathbf{n}\rangle_{\partial K} = 0, \\ -(\mathbf{q}_h, \nabla w)_K + \langle \widehat{\mathbf{q}}_h\cdot\mathbf{n}, w\rangle_{\partial K} = (f,w)_K, \end{aligned} \qquad \forall K \in \mathcal{T}_h. \tag{6}$$

With $\widehat{\mathbf{q}}_h$ commonly referred to as the *numerical flux*, and $\hat{u}_h$ is the *numerical trace*. Similar to other DG variants, $\widehat{\mathbf{q}}_h$ is an approximation to $\mathbf{q}_h$, and we can produce a stable and accurate method by a proper choice of this numerical flux. $\hat{u}_h$ is also an approximation to u_h; however, in our hybrid method, we keep it as a new unknown on the skeleton space. Meanwhile, the prescribed boundary condition on u is applied through $\hat{u}$. Hence, on a given face $e \in \mathcal{E}_h$, we have:

$$\hat{u}_h = \begin{cases} \Pi_\partial g_D & e \in \mathcal{E}_h \cap \partial\Omega_D \\ \lambda_h & e \in \mathcal{E}_h \backslash \partial\Omega_D, \end{cases} \tag{7a}$$

and on $\partial K \in \partial\mathcal{T}_h$, $\widehat{\mathbf{q}}$ is defined as:

$$\widehat{\mathbf{q}}_h = \mathbf{q}_h + \tau(u_h - \hat{u}_h)\mathbf{n}, \tag{7b}$$

where, τ is a *stabilization parameter*, and its corresponding values which can result in a stable method with optimal convergence can be found in the literature [24, 25]. Meanwhile, λ_h is a single-valued approximation to the trace of u on $\mathcal{E}_h\backslash\partial\Omega_D$. Hence, $\hat{u}_h$ is single-valued on any given face in $\mathcal{E}_h$. However, $\widehat{\mathbf{q}}_h$ is not single-valued by construction, and on each of the interior edges we must determine from which element we want to compute it. Therefore, on a given face $e = \partial K^+ \cap \partial K^-$, $\widehat{\mathbf{q}}_h$ has different values on ∂K^+ and ∂K^-. We enforce the conservation of the normal component of $\widehat{\mathbf{q}}_h$ weakly. For this purpose, we require that, for all $e \in \mathcal{E}_h$:

$$\langle [\![\widehat{\mathbf{q}}_h\cdot\mathbf{n}]\!], \mu\rangle_e = \begin{cases} \langle g_N, \mu\rangle_e & e \in \partial\Omega_N \\ 0 & \text{Otherwise} \end{cases} \qquad \forall \mu \in M_h^p(0). \tag{8}$$

Next, we substitute numerical traces from (7) to (6) and sum over all elements. We also want to satisfy the flux conservation condition (8); hence, we are seeking an approximation $(u_h, \mathbf{q}_h, \lambda_h) \in W_h^p \times \mathbf{V}_h^p \times M_h^p(0)$, such that:

$$(\boldsymbol{\kappa}^{-1}\mathbf{q}_h, \mathbf{v})_K - (u_h, \nabla\cdot\mathbf{v})_K + \langle \lambda_h, \mathbf{v}\cdot\mathbf{n}\rangle_{\partial K} = -\langle g_D, \mathbf{v}\cdot\mathbf{n}\rangle_{\partial K\cap\partial\Omega_D}, \qquad \forall K \in \mathcal{T}_h \tag{9a}$$

$$-(\mathbf{q}_h, \nabla w)_K + \langle \mathbf{q}_h\cdot\mathbf{n}, w\rangle_{\partial K} + \langle \tau(u_h - \lambda_h), w\rangle_{\partial K} = (f,w)_K + \langle \tau g_D, w\rangle_{\partial K\cap\partial\Omega_D}, \qquad \forall K \in \mathcal{T}_h \tag{9b}$$

$$\sum_{K\in\mathcal{T}_h} \langle \widehat{\mathbf{q}}_h\cdot\mathbf{n}, \mu\rangle_{\partial K} = \sum_{K\in\mathcal{T}_h} \langle g_N, \mu\rangle_{\partial K\cap\partial\Omega_N}, \tag{9c}$$

for all $(w, \mathbf{v}, \mu) \in W_h^p \times \mathbf{V}_h^p \times M_h^p(0)$. It is worth noting that, (9a, b) are local to each element. One can solve $u_h, \mathbf{q}_h$ in the domain of a given element $K \in \mathcal{T}_h$, assuming that $\hat{u}_h$ is known on ∂K. This forms our so-called *local problem* in the domain of each element. The global equation (9c) can be constructed by summing over all of the elements and substituting the elemental equations; i.e., summing over faces is not required. The Dirichlet boundary condition is applied though equations (9a, b) and the Neumann boundary condition is included in (9c).

Finally, we can derive the following set of equations, which the approximate solutions $(u_h, \mathbf{q}, \lambda_h) \in W_h^p \times \mathbf{V}_h^p \times M_h^p(0)$ should satisfy:

$$\mathscr{a}_K(\mathbf{q}_h, \mathbf{v}) - \mathscr{b}_K(u_h, \mathbf{v}) + \mathscr{c}_K(\lambda_h, \mathbf{v}) = \mathscr{r}_K(\mathbf{v}), \qquad \forall K \in \mathcal{T}_h \tag{10a}$$

$$\mathscr{b}_K^{\mathrm{T}}(\mathbf{q}_h, w) + \mathscr{d}_K(u_h, w) + \mathscr{e}_K(\lambda_h, w) = \mathscr{f}_K(w), \qquad \forall K \in \mathcal{T}_h \tag{10b}$$

$$\sum_{K \in \mathcal{T}_h} \left[\mathscr{c}_K^{\mathrm{T}}(\mathbf{q}_h, \mu) + \mathscr{e}_K^{\mathrm{T}}(u_h, \mu) + \mathscr{h}_K(\lambda_h, \mu) \right] = \sum_{K \in \mathcal{T}_h} \mathscr{l}_K(\mu), \tag{10c}$$

for all $(w, \mathbf{v}, \mu) \in W_h^p \times \mathbf{V}_h^p \times M_h^p(0)$. The bilinear forms and the functionals in the above equations are defined as below:

$$\begin{gathered}
\mathscr{a}_K(\mathbf{q}_h, \mathbf{v}) = (\boldsymbol{\kappa}^{-1}\mathbf{q}_h, \mathbf{v})_K, \quad \mathscr{b}_K(u_h, \mathbf{v}) = -(u_h, \nabla \cdot \mathbf{v})_K, \quad \mathscr{c}_K(\lambda_h, \mathbf{v}) = \langle \lambda_h, \mathbf{v} \cdot \mathbf{n} \rangle_{\partial K}, \\
\mathscr{d}_K(u_h, w) = \langle \tau u_h, w \rangle_{\partial K}, \quad \mathscr{e}_K(\lambda_h, w) = \langle -\tau \lambda_h, w \rangle_{\partial K}, \quad \mathscr{h}_K(\lambda_h, \mu) = \langle -\tau \lambda_h, \mu \rangle_{\partial K}, \\
\mathscr{r}_K(\mathbf{v}) = \langle g_D, \mathbf{v} \cdot \mathbf{n} \rangle_{\partial K \cap \partial \Omega_D}, \quad \mathscr{f}_K(w) = (f, w)_K + \langle \tau g_D, w \rangle_{\partial K \cap \partial \Omega_D}, \quad \mathscr{l}_K(\mu) = \langle g_N, \mu \rangle_{\partial K \cap \partial \Omega_N}.
\end{gathered} \tag{11}$$

3.1. Discretization

Now, consider equation (10) in matrix form:

$$A_K \mathbf{Q}_K - B_K U_K + C_K \Lambda_K = R_K, \qquad \forall K \in \mathcal{T}_h \tag{12a}$$

$$B_K^{\mathrm{T}} \mathbf{Q}_K + D_K U_K + E_K \Lambda_K = F_K, \qquad \forall K \in \mathcal{T}_h \tag{12b}$$

$$C^{\mathrm{T}} \mathbf{Q} + E^{\mathrm{T}} U + H \Lambda = L. \tag{12c}$$

Here, subscript K signifies the element on which we are calculating the unknowns. Moreover, matrices $C^{\mathrm{T}}, E^{\mathrm{T}}, H$ are obtained by assembling the matrices $C_K^{\mathrm{T}}, E_K^{\mathrm{T}}, H_K$ over all elements.

As mentioned before, (12a, b) are local to each element. As a result we can solve (12a) to obtain $\mathbf{Q}_K$ in terms of U_K and Λ_K:

$$\mathbf{Q}_K = A_K^{-1} \left(R_K + B_K U_K - C_K \Lambda_K \right) \tag{13a}$$

Next, we substitute $\mathbf{Q}_K$ from (13a) to (12b) to obtain a relation for U_K in terms of Λ_K:

$$U_K = \left(B_K^{\mathrm{T}} A_K^{-1} B_K + D_K \right)^{-1} \left[-B_K^{\mathrm{T}} A_K^{-1} R_K + (B_K^{\mathrm{T}} A_K^{-1} C_K + E_K) \Lambda_K \right] \tag{13b}$$

Now, we have all of the required equations in the matrix form, i.e. Eqs. (13a, b) and (12c). The solution procedure is as follows:

- Step 1: Use equation (13b) to obtain an expression for U_k in terms of Λ_K. Substitute this U_K into (13a) and obtain an expression for $\mathbf{Q}_K$ in terms of Λ_K.
- Step 2: Assemble the computed U_K, and $\mathbf{Q}_K$ (which are expressed in terms of Λ_K) from the previous step in (12c) to form a global system of equations for Λ_K. Solve this global system, and distribute the obtained Λ_K between corresponding elements.
- Step 3: Use the distributed Λ_K from the previous step in each element to calculate the local values of U_K and $\mathbf{Q}_K$ using (13).

In the above workflow, steps 1 and 3 are local to each element. However, step 2 is a global task. For future reference, we will refer to the computations done in steps 1 and 3 as *local steps*, and step 2 will be called the *global step*.

3.2. Remarks on Local and Global Systems

Although equations (12a, b) are local to each element, the computational effort of solving them is not negligible. For instance, for a fourth order element in three dimensions, A_K is a dense 375×375 matrix, and we need to solve the corresponding system multiple times. Hence, one should consider an efficient technique to solve these local equations. In this regard, it should be noted that A_K is a symmetric positive-definite matrix. Therefore, for solving $\mathbf{Q}_K$ in (13a), an efficient matrix factorization can become handy. Furthermore, in (13b), by an appropriate choice of τ , D_K becomes symmetric positive-semidefinite. Meanwhile, for the term $B_K^{\mathrm{T}} A_K^{-1} B_K$, we have:

$$\langle B_K^{\mathrm{T}} A_K^{-1} B_K\, w, w\rangle = \langle A_K^{-1} B_K\, w, B_K w\rangle > \alpha \|w\|^2,$$

due to $\ker(B_K) = \{\mathbf{0}\}$. Therefore, $\left(B_K^{\mathrm{T}} A_K^{-1} B_K + D_K\right)$ is positive definite, and (13b) is also solvable using efficient matrix factorizations. In addition, another advantage of increasing the weight of local computations in any hybrid method is making it more appropriate for a tailored parallelization.

Regarding the global system in the problems involving anisotropic and non-homogeneous media, the condition number of the global matrix is sensitive to variation in the material properties in the domain. In other words, if we have a medium with large differences in material properties, there is a high chance of getting a computationally ill-conditioned global matrix. In these cases, using Schwarz [26] or multilevel preconditioners [27, 28] can improve the efficiency of the global solver.

4. Local Post-processing

In this section, we employ a simple local post-processing technique which has been also applied to hybrid Raviart-Thomas and hybrid BrezziDouglasMarini methods [29]. A key property of HDG, which justifies the use of this post-processing scheme is the superconvergence of the average of u_h in the domain of each element [24]. This means, for a discretizaton of order $p \geq 1$, $(u_h^\star - u, 1)_K$ converges to zero with order $(p+2)$.

In this element-by-element approach, we want to find $u_h^\star \in W_h^{p+1}$, such that, its restriction to any $K \in \mathcal{T}_h$, satisfies the following:

$$(\nabla u_h^\star, \nabla w)_K = -(\boldsymbol{\kappa}^{-1}\mathbf{q}_h, \nabla w)_K, \qquad \forall w \in \mathcal{P}^{p+1}(K) \tag{14a}$$

$$(u_h^\star, 1)_K = (u_h, 1)_K. \tag{14b}$$

One can observe that the above system of equations is well-defined for solving $(p+2)^d$ unknowns of $u_h^\star$. The first equation reduces to a trivial relation for all $w \in \mathcal{P}^0(K)$, i.e. w being a constant function. Hence, the second equation is added to make the problem well-posed. By applying this post-processing, the computed $u_h^\star$ will converge to u with a rate of $(p+2)$. However, for the case of $p = 0$, the approximation space of u_h consists of only constant functions and $\|u_h - u\|_{L^2(K)}$ converges with $p+1$ rate. Hence, for this case, $u_h^\star$ will also converges to u_h with $p+1$.

The main advantage of using this post-processing over computing u_h via a higher order discretization is changing the computational balance towards local calculations. This means more work with local matrices which have better condition numbers and are much smaller in size. Hence, we gain more stability and more parallelization at the same time.

5. Local Mesh Refinement

In this study, we are interested in using locally refined grids of quadrilateral elements. One of the most common refinement strategies for quadrilateral grids is based on the octree grid structure (Figure 1) [30]. Despite being a highly efficient and flexible refinement strategy, octree grids involve dealing with hanging nodes. Since, we want to build $\hat{u}$'s continuity into its space, hanging nodes will result in a nonconforming skeleton space. Hence, we need to take appropriate measures to make sure that we obtain the correct space for $\hat{u}_h$.

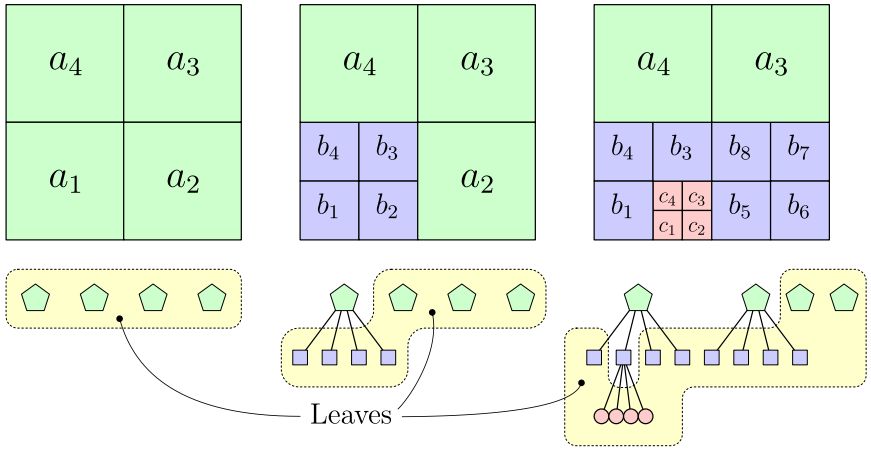


Figure 1: Discretization of the domain Ω by quadrilaterals and the corresponding octrees. Level 0 elements are denoted by a_i, and are shown by pentagons in the tree. Level 1 and 2 elements are shown by squares and circles. respectively.

Among available packages which provide us with an interface to octree grids, we found the deal.II library an efficient and user-friendly software [1]. In this section we present the advantages of using hybrid DG on an octree grid, and how one can make this implementation more efficient. To this end, we first need to briefly present some of the key properties of octree grids. One can find a comprehensive description of this type of grid elsewhere [31]. Next, we explain how by enforcing continuity in the skeleton space for $\hat{u}_h$, we obtain an efficient hybrid method which also has the flexibility of discontinuous Galerkin methods.

5.1. Octree Grid Structure

Let us consider the domain Ω initially discretized with no hanging nodes (e.g. Figure 1). At this stage, all of the elements are at their coarsest level, and they formally have no parents. These elements which are located at level 0 of the corresponding octree are called the "roots". Next we want to refine some of elements based on an appropriate criterion (usually an error indicator). The refinement is done by simply dividing the corresponding reference cells into four (in 2D) and eight cells (in 3D) and mapping the divided reference cell to the real geometry. The resulting refined elements are the children of their coarser parent. At this stage some elements have children and some do not. The elements with no children are called the "leaves". These are also known as "active elements" in deal.II.

Another important rule about octrees that we follow is: "The neighbors of an element can be at most one level coarser or one level finer than the element itself". For example, in Figure 1 we decided to refine element b_2. If we do this refinement without refining a_2, then the grid would have an element of level 0 (a_2) neighboring an element at level 2 (c_2). This will not comply with our rule. As a result, going deeper into a branch of the octree might affect some other branches as well.

5.2. Single-valuedness of $\hat{u}$

In this section, we discuss an efficient approach for constructing a continuous skeleton space for the approximation of $\hat{u}_h$. Let us consider a coarse element neighboring two finer elements; e.g. elements a_2, b_3, and b_2 in Figure 1. We want to define an approximation space inside each of these three elements, such that $(u_h|_K, \mathbf{q}_h|_K) \in \mathcal{P}^p(K) \times \boldsymbol{\mathcal{P}}^p(K)$. These interior spaces can be considered as usual DG spaces without any special treatment. On the other hand, we also want to construct a continuous space $\mathcal{P}^p(e)$ to approximate $\hat{u}_h$ on the common face of these elements. It should be noted that, if we generate this approximation space from the union of two sub-faces on the refined elements, we have to apply extra constraints to guarantee the continuity of the approximation space. However, if we define this space from the coarser side, it will be continuous by construction. Hence, we define the $\hat{u}_h$-space as the span of basis functions of the coarse side. As a result, we do not even store the degrees of freedom of the finer side, and do not need to apply

any constraint equations. Hence, we also decrease the global number of degrees of freedom, and all of the operations for handling the hanging nodes are doable using the basis functions of the coarse side on both sides of the refined faces.

5.3. Refinement Criterion

One of the critical aspects of any method based on adaptive grids is how to pick those elements which require refinement. This can be done by computing an error indicator index for each element, and sort the elements according to this index. Among other refinement criteria, one can find specific a posteriori error estimates for HDG, such as [32]. According to the latter study, for a given element $K \in \mathcal{T}_h$, a function of $\|u_h - \hat{u}_h\|_{L^2(\partial K)}$ and $\|u_h - u_h^\star\|_{L^2(\partial K)}$ can provide an upper and lower bound of the error in each element. Since we are using the deal.II library, another option to obtain an error index in each element is using the available features of this library. The error indicator in this library is based on the a posteriori error analysis proposed in [33]. In this approach, the refinement index for a given element $K \in \mathcal{T}_h$ is calculated via the integral of the jump of the gradient of some characteristic solution (w_h) over ∂K, i.e.: $\|[\![\partial w_h/\partial n]\!]\|_{L^2(\partial K)}$ [34]. By characteristic solution we mean a combination of u, $\mathbf{q}$, their post-processed values, the source term, and the material properties. The common practice for computing this error indicator is using u_h as the characteristic solution (w_h). However, based on a set of numerical experiments on problems with known analytical solutions, we observed that by taking $w_h = |\mathbf{q}_h|$ we can enhance the convergence rates of u_h and $\mathbf{q}_h$. Basically, due to the continuity of $\widehat{\mathbf{q}} \cdot \mathbf{n}$, the gradient of $|\mathbf{q}_h|$ mimics the jump of $\|u_h - \hat{u}_h\|$ across the element borders, which is an error indicator according to [32]. Hence, we have chosen our refinement index based on the gradient of $|\mathbf{q}_h|$. In section 7, we have provided several numerical examples for application of this refinement strategy. In all of these examples, we first compute the solution for an initially defined grid. then, we calculate the jump of the characteristic solution for all elements, and refine the elements with the highest jump values. Thus, we almost double the number of elements in each refinement cycle.

6. Parallel Computing Mechanisms

In order to improve the computational performance of the introduced approach, we have added two layers of parallelism to our implementation. In the first step, we decompose our domain into smaller sub-domains and distribute these sub-domains between our computation cores. Hence, the local and global steps introduced in section (3.1) will be done in parallel. This parallelism strategy, which is known as *distributed memory* parallelism can result in a noticeable speedup in both local and global computations. We have implemented this mechanism using the p4est [3] library included in deal.II. However, in order to use the face counting technique explained in section 5, we have added a set of extra MPI node-to-node communications to transfer the global face numbers between different sub-domains.

In order to improve the efficiency of the computation in each sub-domain, we have also included a second layer of parallelism in our implementation. At this stage, we share the computational tasks in each sub-domain between multiple threads. This is done using OpenMP worksharing directives. It should be noted that, the main computational advantage of HDG is achieving a trade-off between a smaller global system of equations and larger local systems. This trend may be typically achieved by increasing the order of the elements. Hence, we can get the accuracy of a low-order method with large global matrix, by using a high-order method with smaller global matrix. As will be explained in the numerical results section, by increasing the order of elements, the local steps will dominate the overall computational time of the method. Hence, our main motivation for adding the shared memory parallelism is to be able to move towards higher order methods. This way we can reduce the global number of unknowns at the expense of larger local systems.

Despite its advantages, hybrid parallelism has its own downsides as well. One of the main concerns when using multiple threads in combination with distributed parallelism is thread safety. Generally, thread safety is a mechanism which guarantees that if multiple threads want to access the same memory location, the outcome is a defined behavior. In this study, we mainly use PETSc [2] as the solver of our global system. It should be noticed that, for performance reasons PETSc is *not* thread safe. This becomes an important issue when we want to assemble our global matrix using PETSc's `MatSetValues` function. This function can not

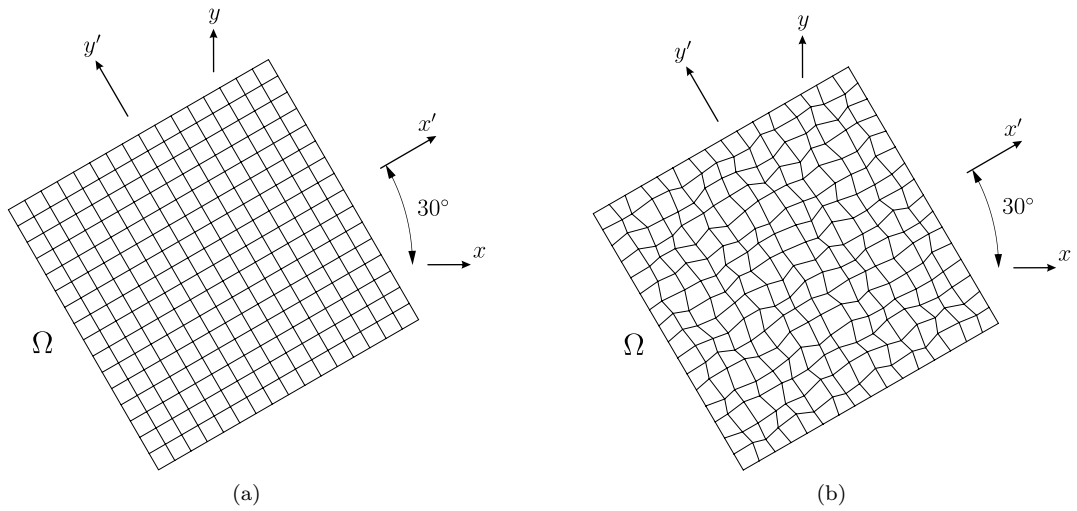


Figure 2: The domain of analysis and (a): a triangulation with regular shaped elements, and (b): a triangulation with randomly distorted elements.

be called by multiple threads at a given time. In order to resolve this issue, we manually apply thread locks in our implementation via OpenMP synchronization directives.

7. Numerical Experiments

In this section, we present a set of numerical experiments to examine the accuracy, convergence properties and performance of the described technique. The first experiment shows the convergence rates of the approximated unknowns for a simple problem. In the second experiment, we consider a simple model problem with a variation of nonhomogeneous and anisotropic material properties. In the last experiment, we solve a source-sink problem, with different material properties. All of these experiments are coded in C++, with the assistance of a few numerical libraries. We have used the deal.II library to handle the triangulation, adaptive refinement, and interfacing with the visualization software. deal.II depends on some other libraries, as well. Among others, for distributed grid it uses the p4est library [3]. As for local calculations inside the elements, we use the Eigen matrix algebra library [35]. Meanwhile, for solving the global system of equations, we use a combination of BoomerAMG preconditioner of Hypre [4] and conjugate gradient solver of PETSc [2]. In all of the experiments we use the default options for the BoomerAMG preconditioner.

Example 1. For the first experiment, we consider the domain of analysis $\Omega \subset \mathbb{R}^2$ to be the square $(-1, 1)^2$ rotated 30° clockwise about its center (cf. Figure 2). We want to solve equation (1), in this domain with $f = 2\pi^2(\mathrm{e}^{x+y} + \mathrm{e}^{x-y})\sin(\pi x)\cos(\pi y) - \pi\mathrm{e}^{x+y}\cos(\pi x)\cos(\pi y) - \pi\mathrm{e}^{x-y}\sin(\pi x)\sin(\pi y)$, $\kappa_{xx} = \mathrm{e}^{x+y}$, $\kappa_{yy} = \mathrm{e}^{x-y}$, and $\kappa_{xy} = 0$. For the first part of this example, we assume a Dirichlet problem with the boundary condition on $\partial\Omega \equiv \partial\Omega_D$ according to: $g_D = \sin(\pi x)\cos(\pi y)$. The analytical solution of this problem is $u = \sin(\pi x)\cos(\pi y)$.

We use quadrilateral grids with 2^n elements in each direction, where $n \in \{1, 2, \cdots, 7\}$. In order to study the effect of irregular shaped elements, we consider two types of meshes. First, we solve the problem with regular square shaped elements and then we apply a random shape distortion on every element. We use elements of order $p = 0, 1, \cdots, 4$ with Legendre modal basis. The convergence rates of $u, q, u^\star$ for the square shaped elements are presented in Table 1. The results are converging with order $p+1$ for u, q, which means they are also converging for $p = 0$ elements. Meanwhile, for all approximation orders except $p = 0$, we can recover an approximate solution $u^\star$ which is converging to u with $p + 2$.

Table 1: The approximation errors and convergence rates of the computed solutions for the grid with regular shaped elements, in example 1.

Order (p)	Num. cells (N)	$\|u-u_h\|_{L^2(\Omega)}$		$\|q-q_h\|_{L^2(\Omega)}$		$\|u-u_h^\star\|_{L^2(\Omega)}$	
		Error	Rate	Error	Rate	Error	Rate
0	$2^4 \times 2^4$	3.31E-01		3.16E+00		2.95E-01	
	$2^5 \times 2^5$	2.05E-01	0.69	1.76E+00	0.84	1.89E-01	0.64
	$2^6 \times 2^6$	1.18E-01	0.79	9.47E-01	0.89	1.11E-01	0.77
	$2^7 \times 2^7$	6.43E-02	0.88	4.96E-01	0.93	6.11E-02	0.86
1	$2^4 \times 2^4$	1.09E-02		1.55E-01		3.15E-03	
	$2^5 \times 2^5$	2.73E-03	1.99	4.09E-02	1.92	4.60E-04	2.78
	$2^6 \times 2^6$	6.87E-04	1.99	1.06E-02	1.94	6.34E-05	2.86
	$2^7 \times 2^7$	1.73E-04	1.99	2.75E-03	1.95	8.43E-06	2.91
2	$2^4 \times 2^4$	3.55E-04		5.59E-03		7.03E-05	
	$2^5 \times 2^5$	4.48E-05	2.98	7.34E-04	2.93	5.07E-06	3.79
	$2^6 \times 2^6$	5.65E-06	2.99	9.58E-05	2.94	3.45E-07	3.88
	$2^7 \times 2^7$	7.10E-07	2.99	1.25E-05	2.93	2.27E-08	3.93
3	$2^4 \times 2^4$	8.86E-06		1.32E-04		1.21E-06	
	$2^5 \times 2^5$	5.57E-07	3.99	8.50E-06	3.95	4.25E-08	4.83
	$2^6 \times 2^6$	3.50E-08	3.99	5.45E-07	3.96	1.42E-09	4.90
	$2^7 \times 2^7$	2.19E-09	4.00	3.50E-08	3.96	4.61E-11	4.95
4	$2^4 \times 2^4$	1.77E-07		2.84E-06		1.88E-08	
	$2^5 \times 2^5$	5.58E-09	4.99	9.26E-08	4.94	3.24E-10	5.86
	$2^6 \times 2^6$	1.75E-10	4.99	3.02E-09	4.94	5.37E-12	5.92
	$2^7 \times 2^7$	5.51E-12	4.99	9.83E-11	4.94	9.02E-14	5.89

Table 2: The approximation errors and convergence rates of the computed solutions for the grid with distorted elements and mixed boundary conditions, in example 1.

Order (p)	Num. cells (N)	$\|u-u_h\|_{L^2(\Omega)}$		$\|q-q_h\|_{L^2(\Omega)}$		$\|u-u_h^\star\|_{L^2(\Omega)}$	
		Error	Rate	Error	Rate	Error	Rate
0	$2^4 \times 2^4$	4.20E-01		3.13E+00		2.75E-01	
	$2^5 \times 2^5$	2.63E-01	0.67	1.77E+00	0.83	1.50E-01	0.88
	$2^6 \times 2^6$	1.53E-01	0.79	9.59E-01	0.88	7.87E-02	0.93
	$2^7 \times 2^7$	8.32E-02	0.88	5.05E-01	0.93	4.05E-02	0.96
1	$2^4 \times 2^4$	1.04E-02		1.63E-01		2.36E-03	
	$2^5 \times 2^5$	2.57E-03	2.01	4.34E-02	1.91	3.35E-04	2.82
	$2^6 \times 2^6$	6.44E-04	2.00	1.13E-02	1.94	4.61E-05	2.86
	$2^7 \times 2^7$	1.61E-04	2.00	2.92E-03	1.95	6.26E-06	2.88
2	$2^4 \times 2^4$	3.36E-04		5.79E-03		4.34E-05	
	$2^5 \times 2^5$	4.22E-05	2.99	7.60E-04	2.93	3.00E-06	3.86
	$2^6 \times 2^6$	5.30E-06	2.99	9.89E-05	2.94	2.01E-07	3.90
	$2^7 \times 2^7$	6.65E-07	2.99	1.28E-05	2.94	1.33E-08	3.92
3	$2^4 \times 2^4$	8.35E-06		1.38E-04		7.11E-07	
	$2^5 \times 2^5$	5.23E-07	4.00	8.93E-06	3.95	2.41E-08	4.88
	$2^6 \times 2^6$	3.28E-08	4.00	5.73E-07	3.96	7.94E-10	4.92
	$2^7 \times 2^7$	2.05E-09	4.00	3.67E-08	3.97	2.61E-11	4.92
4	$2^4 \times 2^4$	1.66E-07		2.92E-06		1.11E-08	
	$2^5 \times 2^5$	5.22E-09	4.99	9.51E-08	4.94	1.86E-10	5.90
	$2^6 \times 2^6$	1.64E-10	4.99	3.08E-09	4.95	3.02E-12	5.94
	$2^7 \times 2^7$	5.11E-12	5.00	9.93E-11	4.96	4.93E-14	5.94

To study the convergence properties of the method in a more general grid, we applied a random distortion on all of the elements of the grid, and solved the same problem as we solved with the regular grid. The corresponding results are presented in table 2. Here, we have applied Neumann boundary condition on the two boundaries with their normals parallel to y', and Dirichlet boundary condition on the other two sides. All of the boundary data are based on the exact solution which are presented for the regular grid. The Neumann boundary condition is applied as: $g_N = \mathbf{q}\cdot\mathbf{n}$, with $q_x = -\pi \mathrm{e}^{x+y}\cos(\pi x)\cos(\pi y)$, $q_y = \pi \mathrm{e}^{x-y}\sin(\pi x)\sin(\pi y)$. Again, we get optimal convergence for $u, \mathbf{q}$, and superconvergence on $u^\star$.

To study the execution time of different stages of computation, the computation time for the most refined mesh of different orders are presented in Table 3. Furthermore, the time balance between local and global computations are also listed in this table. As mentioned in section 3.2, by local computations, we mean the first and third steps of computation. On the other hand, global computation means preconditioning and solving the global system of equations. Here, we have used the BoomerAMG preconditioner from Hypre package, and adjusted the relative tolerance of the global solver to 10^{-12}. These results were obtained on a single node of the Stampede supercomputer in Texas Advanced Computing Center (TACC), which has sixteen cores. For these results no parallel mechanism has been used. This means only one core is performing all of the local and global computation. One can observe that the computation time is considerably high for the relatively small model. In order to examine the performance improvement by using the distributed parallelism, we have performed four extra experiments on the grid with $2^7 \times 2^7$ elements of fourth order. Table 4 lists the results corresponding to these cases. The first row in this table presents the execution time for a single core run, i.e. one sub-domian. This base case, is equivalent to the last row of Table 3. In this table, we have also listed the *computation speedups* corresponding to each case. Speedup characterizes the amount of performance gain using $n \geq 1$ cores, and is defined as follows:

$$S_n = \frac{\text{Execution time using one core}}{\text{Execution time using } n \text{ cores}}.$$

The *computation efficiency* using n cores is defined as $E_n = S_n/n$. This means the closer S_n to n, the more efficient technique we get. Both speedup and efficiency characterize the *scalability* of the method. Obviously, the speedup and efficiency of using a single core is equal to 1.0. Next, we decompose our domain into two parts and use two cores to perform the local and global computations. The relevant speedups in Table 4 show that the local calculations can be scaled very efficiently. However, the global solve step is not as efficient. We continue increasing the number of sub-domains and the results follow the same trend. It is worthwhile noting that, for sixteen cores, we have obtained a speedup of 13.05, which means an efficiency of 81%. However, our local computation efficiency is equal to 99%. As we have mentioned before, this signifies the importance of using higher order elements and increasing the computational balance towards local computations.

Next, we want to evaluate the behavior of the adaptive refinement algorithm. We consider the same problem as explained before, but instead of a global refinement of the grid, we refine the mesh according to the error indicator explained in section 5. In Figure 3, the convergence plots of $(u_h, \mathbf{q}_h)$ for different polynomial orders are presented. Due to various element sizes in the refined grid, we have chosen $1/\sqrt{N}$ as an indicator of the average element size. Because of the uniformity and smoothness of the source term and material properties, one should not expect an optimal convergence from the adaptive grid. We will observe the advantages of the adaptive grids in the heterogeneous examples given below.

Example 2. In this example we examine the solver performance in a three dimensional problem. We consider our domain of analysis $\Omega \subset \mathbb{R}^3$ to be the cube $(-1, 1)^3$. We want to solve the equation (1), with $\kappa_{xx} = e^{x+y}$, $\kappa_{yy} = e^{y+z}$, $\kappa_{zz} = e^{z+x}$, and $f = \pi e^{y+z}\sin(\pi x)\sin(\pi y)\sin(\pi z) + \pi\cos(\pi y)\sin(\pi z)[\pi\sin(\pi x)(e^{x+y} + e^{x+z} + e^{y+z}) - e^{x+y}\cos(\pi x)] - \pi e^{x+z}\sin(\pi x)\cos(\pi y)\cos(\pi z)$. In this example, we consider a Dirichlet problem. The boundary condition on $\partial\Omega \equiv \partial\Omega_D$ is determined according to: $g_D = \sin(\pi x)\cos(\pi y)\sin(\pi z)$. The analytical solution of this problem is $u = \sin(\pi x)\cos(\pi y)\sin(\pi z)$.

We use hexahedral grids with 2^n elements in each direction, where $n \in \{1, 2, \cdots, 6\}$. We assume cube shaped elements (without any distortion). Since, we showed convergence rates for a two dimensional problem, we just mention that, similar to the two dimensional case, both $u_h, \mathbf{q}_h$ will converge with order $p+1$ to u, q,

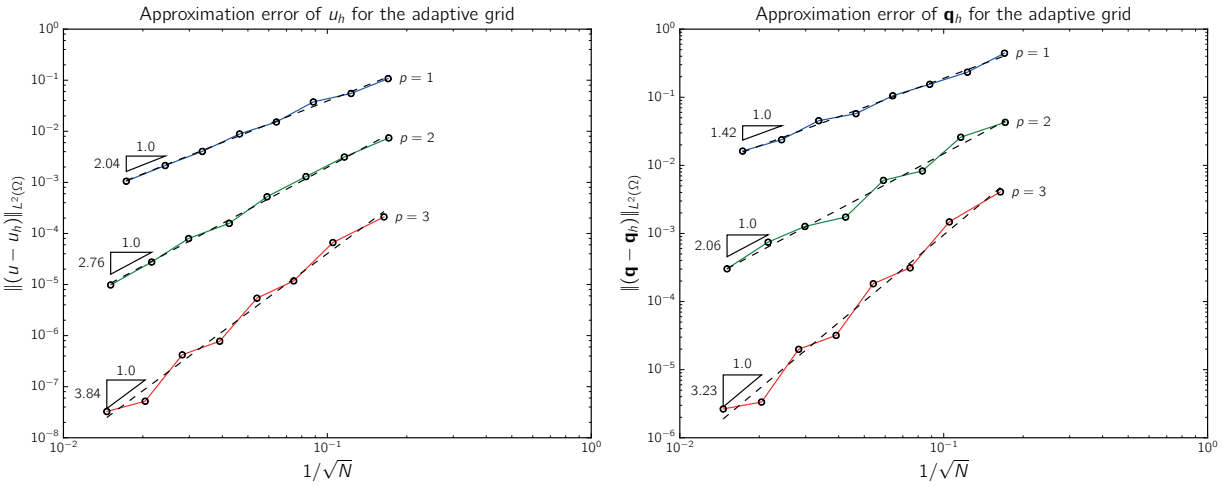


Figure 3: Approximation error of $u^h, \mathbf{q}^h$ in adaptive grids of different order.

Table 3: Number of local and global unknowns in the Dirichlet problem of example 1 for the grid with $2^7 \times 2^7$ elements of different orders, and the time spent for local and global computations. No parallel mechanism is employed.

Order	Num. local unknowns (u_h)	Num. global unknowns ($\hat{u}_h$)	Computation time (s)			
			Local Step 1	Global Solve	Local Step 3	Total
0	16,384	32,512	1.56	0.68	1.36	3.59
1	65,536	65,024	2.74	1.50	2.23	6.47
2	147,456	97,536	5.70	27.46	5.12	38.29
3	262,144	130,048	11.68	40.70	9.45	61.83
4	409,600	162,560	26.12	50.83	21.01	97.95

Table 4: Execution time and speedup of different parallel setups for the grid with $2^7 \times 2^7$ elements of order 4.

Num. Cores	Computation time (s)				Speedup (S_n)			
	Step 1	Global	Step 3	Total	Step 1	Global	Step 3	Total
1	26.12	50.83	21.01	97.95	1.00	1.00	1.00	1.00
2	13.07	30.39	10.55	54.01	2.00	1.67	1.99	1.81
4	6.58	16.13	5.28	27.99	3.97	3.15	3.98	3.50
8	3.25	9.09	2.64	14.99	8.03	5.59	7.96	6.54
16	1.63	4.56	1.32	7.51	16.03	11.15	15.94	13.05

which means they are also converging for $p = 0$ elements. Furthermore, for all approximation orders except $p = 0$, we can recover an approximate post-processed solution $u^\star$ which is converging to u with $p + 2$. In this example, we want to check the execution time of different stages of computation. In Table 5 we have summarized the computation time for the most refined grids of different orders, i.e. $2^6 \times 2^6 \times 2^6$ elements. The time balance between the three steps of analysis discussed in section 3.2 are also listed in this table. Again, we have used the BoomerAMG preconditioner [36], and adjusted the relative tolerance of the global solver to 10^{-12}. These timing results are obtained using 256 cores of the Stampede supercomputer. This means we have decomposed our domain to 256 sub-domains and used one core for each sub-domain. Notice that by increasing the order of elements the computational balance changes towards the local computation. In order to examine the performance improvement by using the distributed parallelism, we have performed two sets of experiments on the grids with elements of order one and four. In Table 6, we have listed the execution time for a grid with $2^7 \times 2^7 \times 2^7$ elements of order 1 using different number of cores. This grid has around 16.5 million local unknowns (u_h) and 25 million global unknowns ($\hat{u}_h$). As explained in the two dimensional problem, the local computation time is again perfectly scaled by increasing the number of active cores. However, the global step does not scale very well. This can be attributed to high communication time in the global solver. Moreover, the local computation has a low share in the total execution time. This results in the low scalability of the method with order 1 elements. However, the results for the grid with fourth order elements show another trend. As we observe in Table 7, the computation weight of local steps for elements of order 4 is relatively high compared to global solve step. Hence, the total scalability of the method is much higher than the case with the first order elements. In this case, we had around 19 million global unknowns ($\hat{u}$), and 32.5 million local unknowns (u). Moreover, for this fourth order grid, even the global solve step shows a very high scalability. It seems that the larger bandwidth of the global matrix corresponding to the high order grid has been handled very well by the parallel preconditioner and the global solver.

Although using high order elements in practical applications where we have discontinuous material properties might seem an unsuitable choice, these type of elements can still be very useful when the material properties are piecewise smooth (which is commonly encountered in practice). In such applications, one can use high order elements in homogeneous areas and use lower order elements in the regions close to jump discontinuities. Thus, we can benefit from the nice properties of different element orders, simultaneously.

Regarding the number of DOFs of HDG compared to a primal DG formulation (such as SIPG), one should note that for the same problem, SIPG has $O(N(p+1)^d)$ DOFs, and HDG has $O(dN(p+1)^{d-1})$ DOFs. Hence, for large N and $p + 1 = d$, the size of the global matrix of SIPG is roghly equal to that of HDG. However, considering the number of nonzero entries in the stiffness matrix (which is the actual work in the solver), we have:

$$N_{\text{HDG}} \coloneqq \text{Nember of nonzero entries in HDG} = O(Nd(p+1)^{d-1} \times (4d-1)(p+1)^{d-1})$$
$$N_{\text{SIPG}} \coloneqq \text{Nember of nonzero entries in SIPG} = O(N(p+1)^{d} \times (2d+1)(p+1)^{d})$$

Hence, for example, for three dimensional case (i.e. $d = 3$) we have:

$$\frac{N_{\text{HDG}}}{N_{\text{SIPG}}} = \frac{33(p+1)^{2d-2}}{5(p+1)^{2d}} = \frac{33}{7(p+1)^2}$$

Thus, for second order elements in three dimensional case, the nonzero entries of HDG is roughly half of the nonzero entries of SIPG. For $p = 3$, this ratio goes down to 30%. Hence, by using hybridized elements with $p > 1$, we can gain a significant boost in performance of the global solver compared to primal DG methods such as SIPG.

Example 3. We have established the accuracy of the employed method in the previous examples. Here, we want to solve two physical problems with nonhomogeneous and anisotropic material properties. We will use the adaptively refined meshes to capture the regions of sharp material change. These jump discontinuities are handled only through quadrature. Due to the high variation of diffusivity through the medium, solving

Table 5: Number of local and global unknowns in the Dirichlet problem of example 2 for the grid with $2^6 \times 2^6 \times 2^6$ elements of different orders, and the time spent for local and global computations. 256 cores were used for this computation.

Order	Num. local unknowns (u_h)	Num. global unknowns ($\hat{u}_h$)	Computation time (s)			
			Step 1	Global	Step 3	Total
0	262,144	774,144	0.74	10.29	0.81	11.84
1	2,097,152	3,096,576	1.31	19.22	1.51	22.04
2	7,077,888	6,967,296	9.71	30.49	10.22	50.52
3	16,777,216	12,386,304	85.93	199.64	72.00	357.57
4	32,768,000	19,353,600	593.64	288.98	388.04	1270.66

Table 6: Execution time and speedup of different parallel setups for the grid with $2^7 \times 2^7 \times 2^7$ elements of order 1.

Cores	Computation time (s)				Speedup (S_n)			
	Step 1	Global	Step 3	Total	Step 1	Global	Step 3	Total
256	1.17E+01	5.62E+01	1.38E+01	8.17E+01	1.00	1.00	1.00	1.00
512	5.95E+00	4.76E+01	6.93E+00	6.05E+01	1.96	1.18	1.99	1.35
1024	3.00E+00	4.24E+01	3.49E+00	4.89E+01	3.9	1.32	3.95	1.67

Table 7: Execution time and speedup of different parallel setups for the grid with $2^6 \times 2^6 \times 2^6$ elements of order 4.

Cores	Computation time (s)				Speedup (S_n)			
	Step 1	Global	Step 3	Total	Step 1	Global	Step 3	Total
256	5.93E+02	2.88E+02	3.88E+02	1.27E+03	1.00	1.00	1.00	1.00
512	3.02E+02	1.48E+02	1.96E+02	6.46E+02	1.96	1.96	1.98	1.96
1024	1.52E+02	7.60E+01	1.01E+02	3.29E+02	3.90	3.79	3.84	3.86

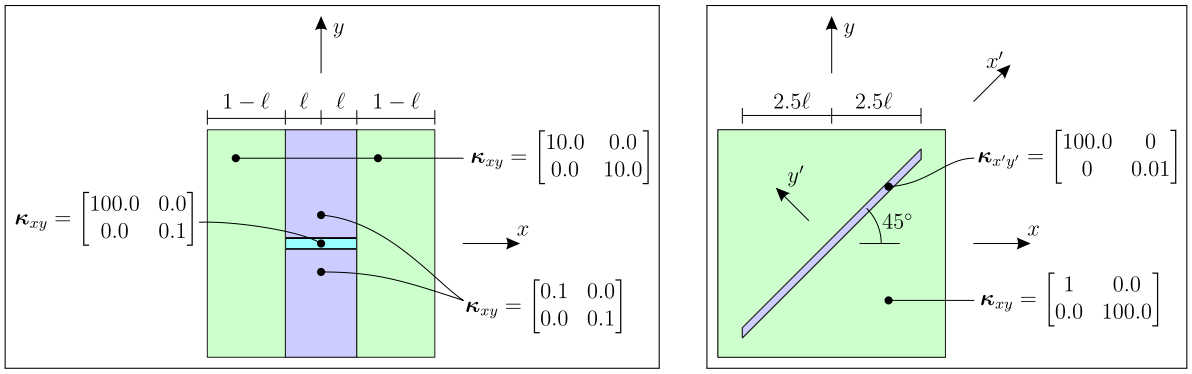


Figure 4: Setups of the problems in the third example ($\ell = \pi/10$); (a): Horizontal pipe problem; (b): Inclined pipe problem.

some of these global systems with iterative methods is not possible without an appropriate preconditioner. Hence, we have used BoomerAMG preconditioner from Hypre package [4, 36] to obtain an efficient solver.

Let us consider two setups as shown in Figure 4. We have a $(-1, 1)^2$ square with no flow boundary condition in its upper and lower edges and Dirichlet boundary condition on the left and right edges with $g_D = 10$ and $g_D = 0$, respectively. The only difference of the two setups is the arrangement of different material properties inside the domain. In the first problem there is a horizontal pipe with anisotropic material with length ℓ and thickness $\ell/5$; here, we have taken length ℓ equal to $\pi/10$. In the second problem, we have a permeable layer which is rotated 45° clockwise. The thickness of this layer is also $\ell/5$. The diffusivity tensor in the rotated coordinates ($\boldsymbol{\kappa}_{x'y'}$) is given in Figure 4. One can obtain ($\boldsymbol{\kappa}_{xy}$) using $\kappa_{ij} = \sum_{i'} \sum_{j'} \beta_{ii'} \beta_{jj'} \kappa_{i'j'}$, with $\beta_{ii'}$ being the cosine of the angle between the i and i' axes. In both problems we start with a grid of $N = 2^3 \times 2^3$ third order elements, and we solve the equations. Then, we use our refinement criterion and solve the equations again for the new locally refined mesh. We continue this iteration until we get a mesh with eight refinement levels.

In the first problem, due to the impermeable layer at the top and bottom of the pipe, one can expect the flow to choose the longer discharge distance and go through the permeable pipe. Hence, one should see a noticeable flow around and inside of the pipe. Meanwhile, since there is almost no flow inside the impermeable material at the top and bottom of the pipe, we should observe a sharp change of flow in this region. The grid structure in the first, fourth and final steps of refinement are shown in Figure 5. Obviously, the adaptive refinement has been able to resolve the material change in this region. The corresponding plots of u_h and $|\mathbf{q}_h|$ are also shown in this Figure. One can see the sharp change of flow inside the pipe, compared to the impermeable material around it. Furthermore, there is almost no drop in the head at the two ends of the permeable pipe.

In the second problem (Figure 4-b), we have set the vertical permeability (κ_{yy}) of the material around the pipe a hundred times higher than its horizontal permeability (κ_{xx}). Meanwhile, the permeability of the pipe in its longitudinal direction is also ten thousand times higher than its cross-sectional direction. As a result, we expect to see the flow from the left boundary to go down in the y-direction, and enter the pipe and after following the pipe's path, discharge through the right boundary. We should also see high flow gradient at the two ends of the pipe. Meanwhile, there should be a sharp change of flow in the area around the pipe. The adaptive refinement algorithm that we use here is similar to the first problem in this example. The evolution of the grid is shown in Figure 6. Furthermore, the contour plot of the x component of $\mathbf{q}_h$ in the whole domain and vector plot of $\mathbf{q}_h$ at the pipe entrance are shown for the corresponding levels of refinement. As in the previous problem, the sharp change of material is captured by the adaptive refinement, and the high gradient of the flow around the pipe and at its two ends are visible. Furthermore, a large flow in the y-direction is visible at the entrance of the pipe. This is due to the lower permeability of the material in the x-direction compared to the y-direction.

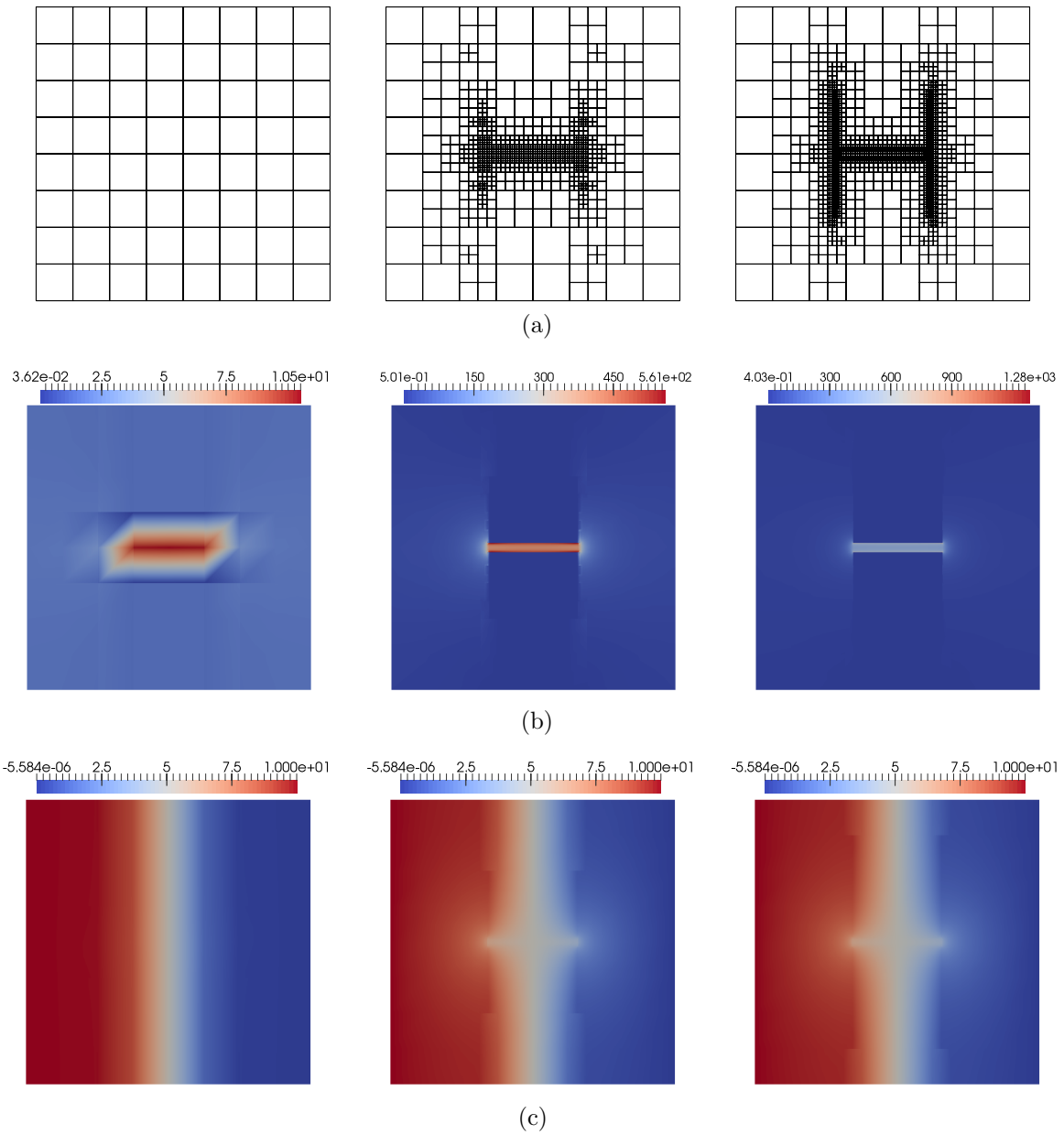


Figure 5: Results for the horizontal pipe problem. (a): Three stages of the evolution of the adaptively refined grid; Plots of (b): $|\mathbf{q}_h|$ and (c):u_h, corresponding to the above grids.

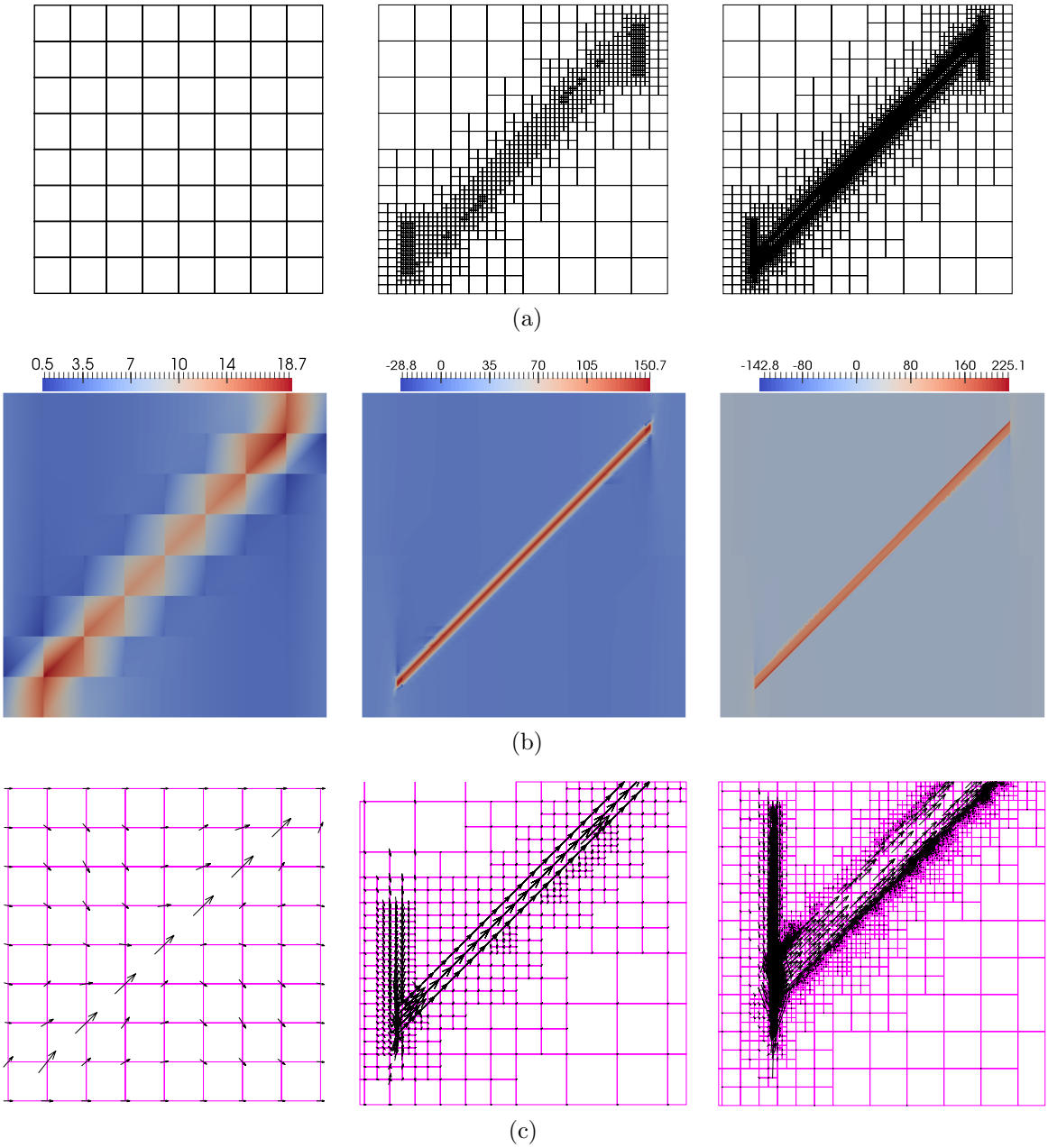


Figure 6: Results for the inclined pipe problem. (a): Three stages of the evolution of the adaptively refined grid; Plots of (b): x component of $\mathbf{q}_h$ and (c):vector plots of $\mathbf{q}_h$ zoomed at the pipe entrance (The vector scales are different for each grid).

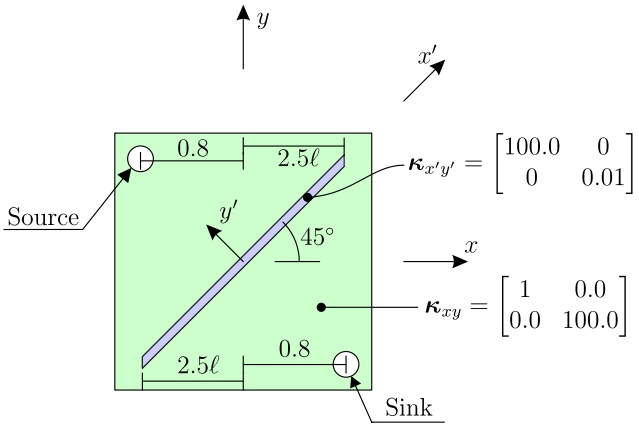


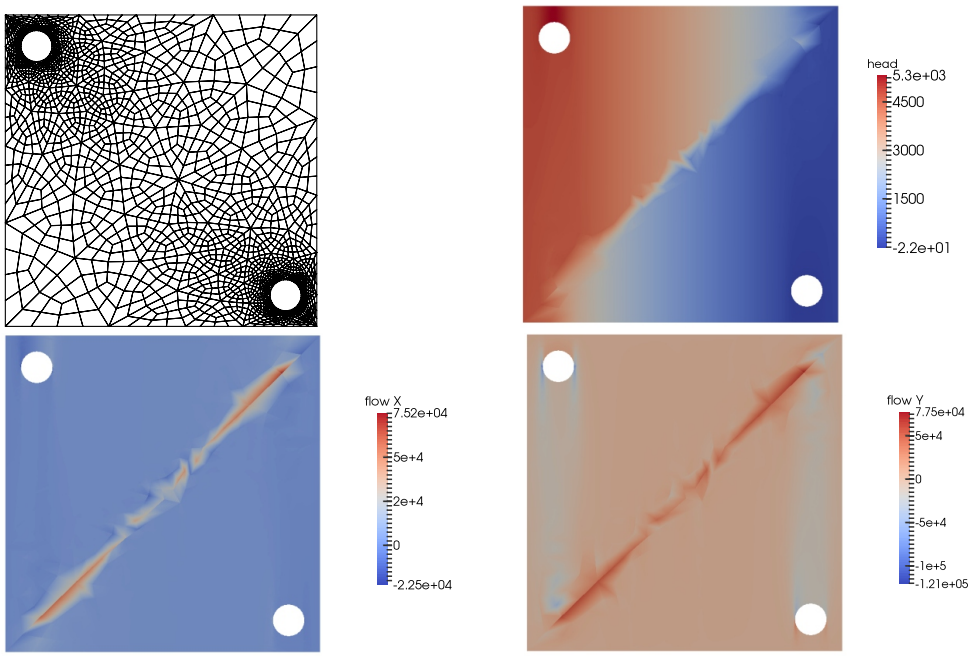
Figure 7: Setup of the problems in the fourth experiment (with $\ell = \pi/10$).

Example 4. In this example, we will solve a two dimensional sink-source type of problem. The problem setup is shown in Figure 7. We have used Gmsh software [37] to produce the computational grid for this problem. All of the material properties and their arrangement are similar to the second part of example 3. As in Example 3, jump discontinuities are handled only through quadrature. In this problem, all of the boundary conditions on the four edges of the square domain are no flow boundaries, and we have added two holes with radius 0.1 inside the domain. The upper left hole is the source hole, with Neumann inflow boundary condition $g_N = 500$. The lower right hole is the sink, with Dirichlet boundary condition $g_D = -10$. Similar to the previous example, the material around the pipe has a low x-direction permeability, and makes the flow from the source hole to go vertically in the y-direction towards the permeable pipe. The flow then follows the pipe's path and at the end of the pipe takes the vertical direction down to the sink. As a result, we should see a sharp change of flow around the pipe, and high gradients near the pipe's two ends. The first grid used in this experiment is shown in Figure 8-a. All of the elements in this grid are second order elements. The calculated values for u_h, and $\mathbf{q}_h$ are also shown in Figure 8-a. Since, the original mesh in this example is more refined than the ones used in the previous examples, we can see the effect of highly permeable pipe in the plots of $\mathbf{q}_x$, and $\mathbf{q}_y$, more clearly. After obtaining the solution on the original grid, we use the same adaptive refinement strategy as explained before, and solve the problem again. We continue this process for five cycles. The grids in the third and fifth steps of refinement are shown in Figure 8-b,c.

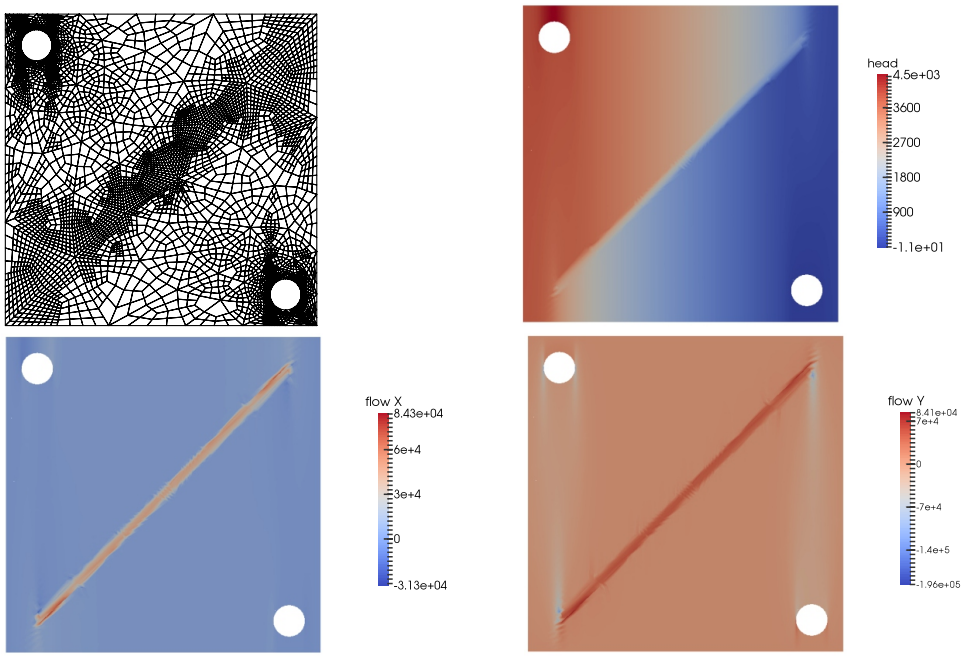
By looking at the plots of the last stage of refinement, we can see a noticeable flow in the y-direction around the left and right side of the source hole. This mainly shows that the anisotropy of material around the source hole has caused the flow to choose the vertical path into the pipe and then to the sink. A similar high y-direction flow is also visible around the two ends of the pipe.

As mentioned before, the high non-homogeneity and anisotropy in this problem has been successfully captured by the adaptive refinement. However, the use of an efficient preconditioner for solving these kind of problems is inevitable. In order to study the performance of the global solver, we have listed the condition number of the preconditioned global matrix, along with the number of iterations of the conjugate gradient method for solving the global system in Table 8. It should be mentioned that the performance of the preconditioner is not up to the expectations. We are using the default options of the Hypre's BoomerAMG, which has not resulted in an efficient technique. Improving the performance of the parallel preconditioner to obtain a more efficient solver is part of our ongoing study. For the current preconditioner, we notice that by increasing the model size, the general trend of the condition number is increasing.

Example 5. In the last example we will solve a three dimensional problem with nonhomogeneous and anisotropic materials. The problem setup is shown in Figure 9. Here, we have a cubic domain, $\Omega \equiv (-1, 1)^3$, with a pipe from one of its corners to the opposite corner. The pipe's radius is $r_{\text{pipe}} = \ell/5$, and it extends from $x = -2.5\ell$ to $x = 2.5\ell$, where $\ell = \pi/10$. The pipe is highly permeable in x' direction and has lower



(a)

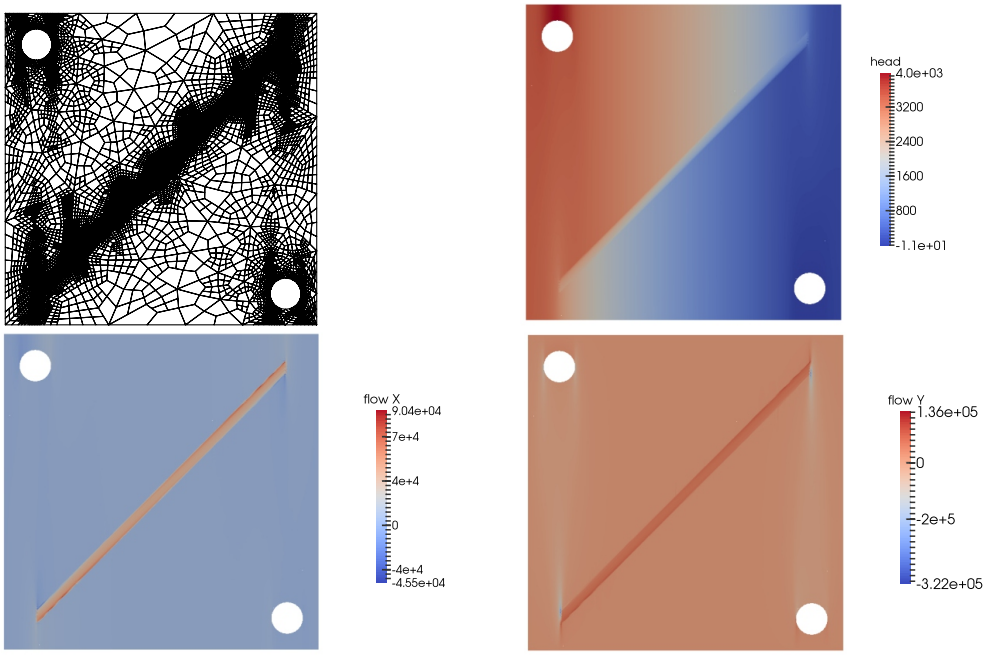


(b)

Figure 8: Results for the sink-source problem; the evolution of the grid, calculated head (u) and the components of the flow vector $\mathbf{q}$; (a): For the original grid; (b): For the 2nd refinement stage;

Table 8: The condition number of the global matrix in example four, and the number of performed iterations for solving the global system.

Refinement level	Num. DOFs	Condition number	Num. iterations
1	13,944	4.77E+04	391
2	25,692	7.13E+04	476
3	48,102	1.01E+05	568
4	91,347	1.38E+05	714
5	173,474	2.00E+05	920



(c)

Figure 8 (Cont.): Results for the sink-source problem; the evolution of the grid, calculated head (u) and the components of the flow vector $\mathbf{q}$; (c): For the last stage of refinement.

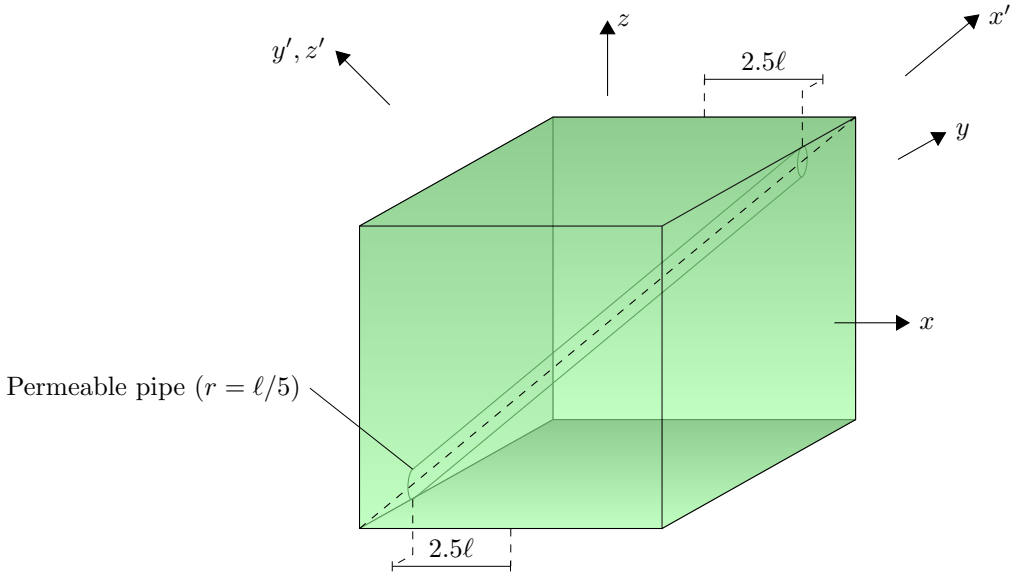


Figure 9: Geometry of the domain in example 5, with $\ell = \pi/10$. x, y, and z are normal to the faces of the cube, and x' is in the direction of the pipe.

permeability in the y' and z' direction, i.e.: $\kappa_{x'x'} = 10$, $\kappa_{y'y'} = 0.1$, $\kappa_{z'z'} = 0.1$. In order to obtain the diffusivity tensor in xyz coordinates, one can use the same relation as we used in the third example, i.e.: $\kappa_{ij} = \sum_{i'} \sum_{j'} \beta_{ii'} \beta_{jj'} \kappa_{i'j'}$. This relation can also be written in the matrix form as: $\boldsymbol{\kappa} = \boldsymbol{R}^{\mathrm{T}} \boldsymbol{\kappa}' \boldsymbol{R}$, where $\boldsymbol{R}$ is the rotation matrix from xyz system to $x'y'z'$. This rotation matrix can be obtained by first rotating the xyz system 45° about the z-axis to reach $x''y'z$, then rotate this new system about y' to get $x'y'z'$. Hence, $\boldsymbol{R}$ can be written as:

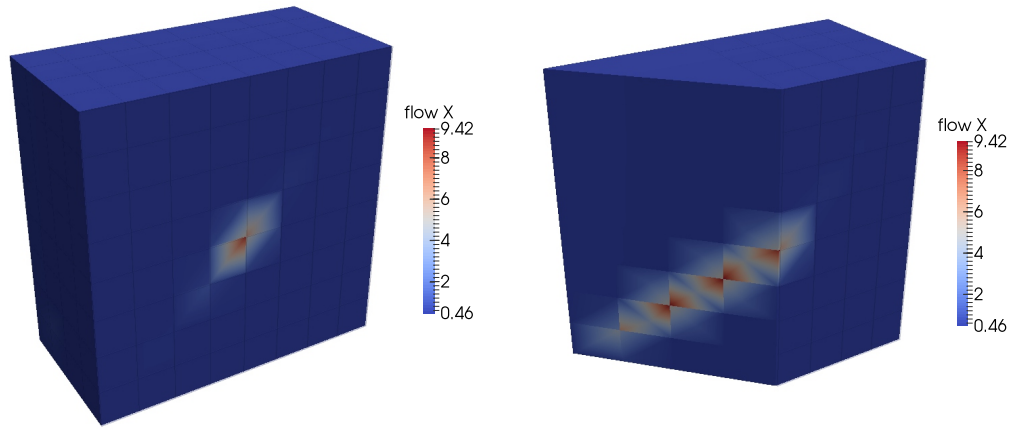
$$\boldsymbol{R} = \begin{bmatrix} \sqrt{6}/3 & 0 & \sqrt{3}/3 \\ 0 & 1 & 0 \\ -\sqrt{3}/3 & 0 & \sqrt{6}/3 \end{bmatrix} \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{3}/3 & \sqrt{3}/3 & \sqrt{3}/3 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -\sqrt{6}/6 & -\sqrt{6}/6 & \sqrt{6}/3 \end{bmatrix}$$

On the face with $x = -1$, we apply Dirichlet boundary condition with $g_D = 10$, and on $x = 1$, we apply $g_D = 0$. We apply no flow boundary condition on all other faces of the cube. Similar to the previous examples, we use the BoomerAMG preconditioner [4, 36] to obtain an efficient parallel solver.

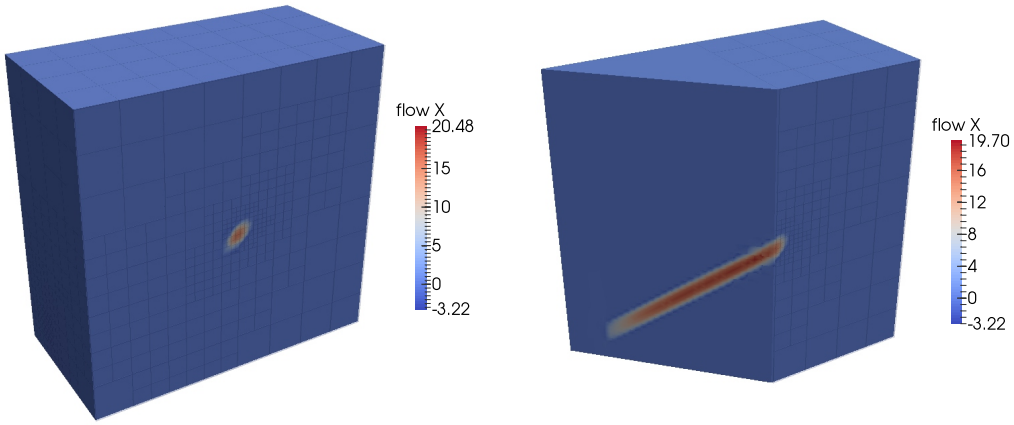
We start with a grid of $2^3 \times 2^3 \times 2^3$ elements of order 2. Then we solve the problem and use our refinement strategy, and solve the problem in the locally refined mesh again. We continue this approach for 6 refinement cycles. The plots of the flow in the x direction for the refinement cycles 0, 3, and 6 are shown in Figure 10. In these plots, we have first clipped the volume with the plane $x = 0$. For each refinement stage, these plots are shown in the left column. Then, we clip this new volume with the plane $x - y = 0$. The corresponding plots are shown in the right column of this figure. The maximum flows in x, y, and z directions are also summarized in Table 9. Once more, we observe that, the adaptive refinement has successfully resolved the areas with material discontinuity which are demanding the highest mesh resolution.

8. Conclusions

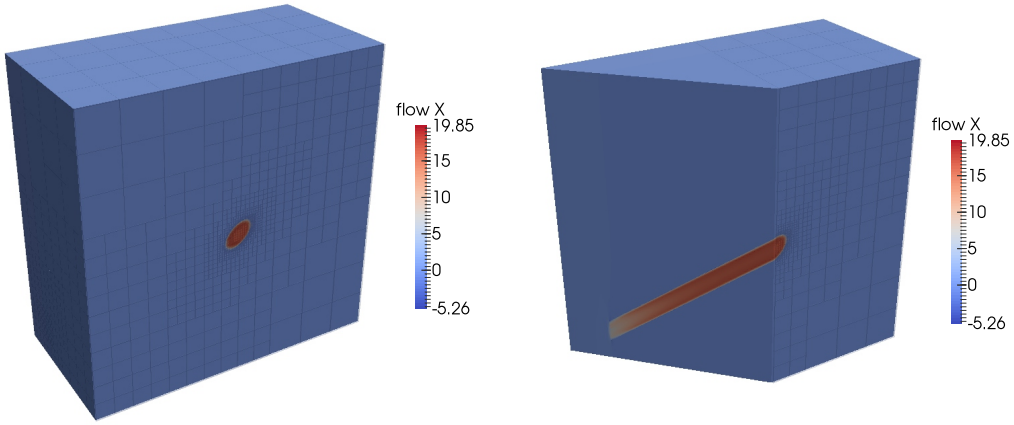
In this study, we have used the hybrid discontinuous Galerkin method to solve the diffusion equation in nonhomogeneous, anisotropic media. Using a set of numerical experiments, it was shown that for a grid with pth order elements, the solution converges at a rate of $p + 1$ for both u_h and $\mathbf{q}_h$ in L^2 norm. We used a simple local post-processing scheme to get a super-convergent solution ($u_h^\star$) for the scalar variable in case of $p \geq 1$. The method is locally conservative and its order can be increased arbitrarily.



(a)



(b)



(c)

Figure 10: Variation of q_x in three cycles of mesh refinement in example 5. (a): cycle 0 (no refinement), (b): cycle 3, and (c): cycle 6.

Table 9: Maximum flows in x, y, and z directions, for different cycles of refinement in example 5.

Refinement level	Num. Elems	q_x	q_y	q_z
0	512	9.42	6.85	6.85
1	1,240	15.76	13.35	13.35
2	3,102	35.02	36.01	36.01
3	8,030	20.48	20.30	19.79
4	21,183	20.56	37.05	37.05
5	55,581	18.79	47.37	47.37
6	151,054	19.85	88.62	88.62

In order to resolve the sharp material interfaces, which requires a high element concentration to capture, we have used an adaptively refined grid. This way, we avoid a uniform grid that unnecessarily increases the number of degrees of freedom.

Due to the large size of the problems in the relevant applications, the best option for solving the global system of equations is iterative solvers. Despite their advantages, using this class of solvers in highly non-homogeneous and anisotropic media requires a suitable preconditioner. In this study we used the conjugate gradient method in combination with an algebraic multigrid preconditioner to enhance the performance of the global solver. Furthermore, the efficiency of the proposed method has been improved by using shared and distributed memory parallelism. The local stages of the computation has been shown to be completely scalable by increasing the number of computation cores. Meanwhile, we have achieved a good scalability in the global solve step.

Another important observation which we have noticed in our numerical experiments was the advantages of the higher order elements in terms of efficiency and accuracy. If we have a piecewise continuous material properties in our domain (which is usually encountered in practical applications), we can use high order elements in smooth material regions to increase the computational weight of the local steps. Since local steps can be scaled linearly with the number of cores, we can achieve a higher scalability using high order grids. Moreover, by using these elements, we can reach accuracy levels which require excessively finer grids for low order simulations.

Acknowledgments

This material is based upon work supported in part by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Award Number DE-SC0009286 as part of the DiaMonD Multifaceted Mathematics Integrated Capability Center, and NSF Grant ACI-1339801. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results. They also acknowledge the support of XSEDE grant TG-DMS080016N.

Bibliography

[1] Wolfgang Bangerth and Guido Kanschat. Concepts for object-oriented finite element software-the deal. ii library. 1999.

[2] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.6, Argonne National Laboratory, 2015.

[3] Carsten Burstedde, Lucas C Wilcox, and Omar Ghattas. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[4] Robert Falgout and Ulrike Yang. hypre: A library of high performance preconditioners. *Computational ScienceICCS 2002*, pages 632–641, 2002.

[5] Bernardo Cockburn, Jayadeep Gopalakrishnan, and Raytcho Lazarov. Unified hybridization of Discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis*, 47(2):1319–1365, 2009.

[6] D.N. Arnold and F. Brezzi. Mixed and nonconforming finite element methods: Implementation, postprocessing and error estimates. *RAIRO Modèl. Math. Anal. Numèr*, 19:7–32, 1985.

[7] P. Castillo, B. Cockburn, I. Perugia, and D. Schötzau. An a priori error analysis of the local discontinuous galerkin method for elliptic problems. *SIAM Journal on Numerical Analysis*, 38:1676–1706, 2000.

[8] B. Riviére, M.F. Wheeler, and V. Girault. A priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems. *SIAM Journal on Numerical Analysis*, 39:902–931, 2001.

[9] B. Cockburn and C.-W. Shu. The local discontinuous galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, 35:2440–2463, 1998.

[10] C. Dawson, S. Sun, and M.F. Wheeler. Compatible algorithms for coupled flow and transport. *Computer Methods in Applied Mechanics and Engineering*, 193:2565–2580, 2004.

[11] James Hyman, Mikhail Shashkov, and Stanly Steinberg. The numerical solution of diffusion problems in strongly heterogeneous non-isotropic materials. *Journal of Computational Physics*, 132(1):130–148, 1997.

[12] J Hyman, J Morel, M Shashkov, and Stanly Steinberg. Mimetic finite difference methods for diffusion equations. *Computational Geosciences*, 6(3-4):333–352, 2002.

[13] Franco Brezzi, Konstantin Lipnikov, and Mikhail Shashkov. Convergence of the mimetic finite difference method for diffusion problems on polyhedral meshes. *SIAM Journal on Numerical Analysis*, 43(5):1872–1896, 2005.

[14] Franco Brezzi, Konstantin Lipnikov, and Mikhail Shashkov. Convergence of mimetic finite difference method for diffusion problems on polyhedral meshes with curved faces. *Mathematical Models and Methods in Applied Sciences*, 16(02):275–297, 2006.

[15] Yu Kuznetsov, K Lipnikov, and M Shashkov. The mimetic finite difference method on polygonal meshes for diffusion-type problems. *Computational Geosciences*, 8(4):301–324, 2004.

[16] Lourenço Beirão da Veiga and Gianmarco Manzini. A higher-order formulation of the mimetic finite difference method. *SIAM Journal on Scientific Computing*, 31(1):732–760, 2008.

[17] A. Younes and V. Fontaine. Hybrid and multi-point formulations of the lowest-order mixed methods for darcy's flow on triangles. *International Journal for Numerical Methods in Fluids*, 58:1041–1062, 2008.

[18] I. Aavatsmark. An introduction to multipoint flux approximations on quadrilateral grids. *Computational Geosciences*, 6:404–432, 2002.

[19] M.G. Edwards and C.F. Rogers. Finite volume discretization with imposed flux continuity for the general tensor pressure equation. *Computational Geosciences*, 2:259–290, 1998.

[20] I. Aavatsmark, G.T. Eigestad, B.T. Mallison, and J.M. Nordbotten. A compact multipoint flux approximation method with improved robustness. *Numerical Methods for Partial Differential Equations*, 24:1329–1360, 2008.

[21] S.F. Matringe, R. Juanes, and H.A. Tchelepi. Mixed finite element and related control volume discretizations for reservoir simulation on three-dimensional unstructured grids. *SPE Reservoir Simulation Symposium Proceedings*.

[22] I. Aavatsmark, G.T. Eigenstad, B.O. Heimsund, and B.T. Mallison. A new finite-volume approach to efficient discretization on challenging grids. *SPE Journal*, 15(3):658–669, 2010.

[23] G. Singh and M.F. Wheeler. Compositional flow modeling using multi-point flux mixed finite element method. *14th European Conference on the Mathematics of Oil Recovery, ECMOR 2014*, 2014.

[24] Bernardo Cockburn, Bo Dong, and Johnny Guzmán. A superconvergent ldg-hybridizable galerkin method for second-order elliptic problems. *Mathematics of Computation*, 77(264):1887–1916, 2008.

[25] Bernardo Cockburn, Johnny Guzmán, and Haiying Wang. Superconvergent discontinuous galerkin methods for second-order elliptic problems. *Mathematics of Computation*, 78(265):1–24, 2009.

[26] Jayadeep Gopalakrishnan. A schwarz preconditioner for a hybridized mixed method. *Computational Methods in Applied Mathematics Comput. Methods Appl. Math.*, 3(1):116–134, 2003.

[27] Zhangxin Chen, Richard E Ewing, Raytcho D Lazarov, Serguei Maliassov, and Yuri A Kuznetsov. Multilevel preconditioners for mixed methods for second order elliptic problems. *Numerical linear algebra with applications*, 3(5):427–453, 1996.

[28] Owe Axelsson and Alexander Padiy. On the additive version of the algebraic multilevel iteration method for anisotropic elliptic problems. *SIAM Journal on Scientific Computing*, 20(5):1807–1830, 1999.

[29] Rolf Stenberg. Postprocessing schemes for some mixed finite elements. *RAIRO-Modélisation mathématique et analyse numérique*, 25(1):151–167, 1991.

[30] David P Young, Robin G Melvin, Michael B Bieterman, Forrester T Johnson, Satish S Samant, and John E Bussoletti. A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics. *Journal of Computational Physics*, 92(1):1–66, 1991.

[31] Tiankai Tu, David R O'Hallaron, and Omar Ghattas. Scalable parallel octree meshing for terascale applications. In *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, pages 4–4. IEEE, 2005.

[32] Bernardo Cockburn and Wujun Zhang. A posteriori error estimates for hdg methods. *Journal of Scientific Computing*, 51(3):582–607, 2012.

[33] DW Kelly, De SR Gago, OC Zienkiewicz, I Babuska, et al. A posteriori error analysis and adaptive processes in the finite element method: Part ierror analysis. *International journal for numerical methods in engineering*, 19(11):1593–1619, 1983.

[34] W. Bangerth and R. Rannacher. Finite element approximation of the acoustic wave equation: Error control and mesh adaptation. *East–West J. Numer. Math.*, 7(4):263–282, 1999.

[35] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.
[36] Van Emden HENSON and Ulrike Meier YANG. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied numerical mathematics*, 41(1):155–177, 2002.
[37] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.