SIAM J. SCI. COMPUT. © 2019 Society for Industrial and Applied Mathematics
Vol. 41, No. 5, pp. S172–S195

UNIFIED GEOMETRIC MULTIGRID ALGORITHM FOR HYBRIDIZED HIGH-ORDER FINITE ELEMENT METHODS[*]

TIM WILDEY[†], SRIRAMKRISHNAN MURALIKRISHNAN[‡], AND TAN BUI-THANH[§]

Abstract. We consider a standard elliptic partial differential equation and propose a geometric multigrid algorithm based on Dirichlet-to-Neumann (DtN) maps for hybridized high-order finite element methods. The proposed unified approach is applicable to any locally conservative hybridized finite element method including multinumerics with different hybridized methods in different parts of the domain. For these methods, the linear system involves only the unknowns residing on the mesh skeleton, and constructing intergrid transfer operators is therefore not trivial. The key to our geometric multigrid algorithm is the physics-based energy-preserving intergrid transfer operators which depend only on the fine scale DtN maps. Thanks to these operators, we completely avoid upscaling of parameters and no information regarding subgrid physics is explicitly required on coarse meshes. Moreover, our algorithm is agglomeration-based and can straightforwardly handle unstructured meshes. We perform extensive numerical studies with hybridized mixed methods, hybridized discontinuous Galerkin methods, weak Galerkin methods, and hybridized versions of interior penalty discontinuous Galerkin methods on a range of elliptic problems including subsurface flow through highly heterogeneous porous media. We compare the performance of different smoothers and analyze the effect of stabilization parameters on the scalability of the multigrid algorithm.

Key words. iterative solvers, geometric multigrid, hybridized methods, multinumerics, high-order, Dirichlet-to-Neumann maps

AMS subject classifications. 65N30, 65N55, 65N22, 65N12, 65F10

DOI. 10.1137/18M1193505

1. Introduction.

Hybridization of finite element methods was first introduced in 1965 [25] for solving linear elasticity problems. Hybridized finite element methods have come a long way since then and a vast amount of research has been done over the past few decades (see, e.g., [10, 19, 20, 28, 39]). Hybridized methods offer a number of significant advantages over the original ones, some of which are (1) the resulting linear system can be significantly smaller and sparser [13, 20]; (2) *hp*-adaptivity is natural for hybridized methods due to the already available skeletal space; (3) it offers a natural way to couple different numerical methods in different parts of the domain (multinumerics) and hence exploit their individual strengths; and (4) for certain problems the trace unknowns can be used to postprocess the solution to obtain superconvergence [1, 20].

[*]Received by the editors June 11, 2018; accepted for publication (in revised form) April 22, 2019; published electronically October 29, 2019.
https://doi.org/10.1137/18M1193505

Funding: The work of the first author was supported by the Office of Science Early Career Research Program. The work of the second and third authors was supported by DOE grant DE-SC0018147 and NSF grant NSF-DMS1620352. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

[†]Center for Computing Research Sandia National Laboratories, Albuquerque, NM 87185 (tmwilde@sandia.gov).

[‡]Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, Austin, TX 78712 (sriramkrishnan@utexas.edu).

[§]Department of Aerospace Engineering and Engineering Mechanics, and Oden Institute for Computational Engineering and Sciences, University of Texas at Austin, Austin, TX 78712 (tanbui@ices.utexas.edu).

Hybridized methods for second-order elliptic equations have been successfuly applied for many practical applications arising in modeling of flow through porous media and are often the methods of choice for these simulations due to their local conservation property [24, 27, 34, 40, 46]. The main challenge facing hybridized methods in these large scale simulations is the construction of scalable solvers for the linear system involving only trace unknowns on the mesh skeleton. Over the past 30 years, a tremendous amount of research has been devoted to the convergence of multigrid methods for such linear systems, both as iterative methods and as preconditioners for Krylov subspace methods. Optimal convergence with respect to the number of unknowns is usually obtained under mild elliptic regularity assumptions [5, 6, 7]. Multigrid algorithms have also been developed for mortar domain decomposition methods [2, 47] and several approaches have been proposed for hybridized mixed finite element methods [8, 15]. Most of these are based on an equivalence between the interface operator and a nonconforming finite element method, and for these methods optimal convergence has already been established [3, 9]. Multigrid algorithms based on restricting the trace (skeletal) space to linear continuous finite element space has been proposed for hybridized mixed methods [29], hybridized discontinuous Galerkin methods [18], and weak Galerkin methods [14]. These algorithms fall under the noninherited category, i.e., the coarse scale operators do not inherit all the properties of the fine scale ones. To the best of our knowledge, no multigrid algorithm has been developed for the case of multinumerics.

The objective of this work is to develop a multigrid algorithm that applies to both hybridized formulations and multinumerics. The algorithm applies to both structured and unstructured grids. At the heart of our approach is the energy-preserving intergrid transfer operators which are a function of only the fine scale Dirichlet-to-Neumann (DtN) maps (to be discussed in detail in section 3). As such they avoid any explicit upscaling of parameters and at the same time allow for the use of multinumerics throughout the domain. The multigrid algorithm presented in this paper thus differs from the existing approaches in that the Galerkin coarse grid operator is a discretized DtN map on every level.

This paper is organized as follows. Section 2 introduces the model problem, notation, and hybridized methods considered in this paper. In section 3, we define the necessary ingredients for our geometric multigrid algorithm, that is, the coarsening strategy, the intergrid transfer operators, the local correction operator, and the smoothing operator. These operators are then used to define the multigrid algorithm. Section 4 presents several numerical examples to study the robustness of the proposed algorithm for different hybridized methods, smoothers, and test cases. Finally, section 5 summarizes our findings and discusses future research directions.

2. Model problem, notation, and hybridized methods.

In this section we will first introduce the notation and discuss some of the locally conservative hybridized methods proposed in recent years for the second-order elliptic equation given by

$$-\nabla \cdot (\mathbf{K}\nabla q) = f \quad \text{in} \quad \Omega, \tag{2.1a}$$

$$q = g_D \quad \text{on} \quad \partial\Omega, \tag{2.1b}$$

where Ω is an open, bounded, and connected subset of $\mathbb{R}^d$, with $d \in \{2,3\}$. Here, $\mathbf{K}$ is a symmetric, bounded, and uniformly positive definite tensor, $f \in L^2(\Omega)$, and $g_D \in H^{3/2}(\partial\Omega)$. Let $\mathcal{T}_h$ be a conforming partition of Ω into N_T nonoverlapping elements T_j, $j = 1, \dots, N_T$, with Lipschitz boundaries such that $\mathcal{T}_h := \cup_{j=1}^{N_T} T_j$ and $\overline{\Omega} = \overline{\mathcal{T}_h}$. The mesh size h is defined as $h := \max_{j\in\{1,\dots,N_T\}} \operatorname{diam}(T_j)$. We denote the skeleton of the mesh by $\mathcal{E}_h := \cup_{j=1}^{N_T} \partial T_j$: the set of all (uniquely defined) interfaces e between elements. We conventionally identify $\mathbf{n}^-$ as the outward normal vector on the boundary ∂T of element T (also denoted as T^-) and $\mathbf{n}^+ = -\mathbf{n}^-$ as the outward normal vector of the boundary of a neighboring element (also denoted as T^+). Furthermore, we use $\mathbf{n}$ to denote either $\mathbf{n}^-$ or $\mathbf{n}^+$ in an expression that is valid for both cases, and this convention is also used for other quantities (restricted) on a face $e \in \mathcal{E}_h$.

For simplicity, we define $(\cdot,\cdot)_T$ as the L^2-inner product on a domain $T \subset \mathbb{R}^d$ and $\langle\cdot,\cdot\rangle_T$ as the L^2-inner product on a domain T if $T \subset \mathbb{R}^{d-1}$. We shall use $\|\cdot\|_T := \|\cdot\|_{L^2(T)}$ as the induced norm for both cases. Boldface lowercase letters are conventionally used for vector-valued functions and in that case the inner product is defined as $(\mathbf{u},\mathbf{v})_T := \sum_{i=1}^m (\mathbf{u}_i,\mathbf{v}_i)_T$, and similarly $\langle\mathbf{u},\mathbf{v}\rangle_T := \sum_{i=1}^m \langle\mathbf{u}_i,\mathbf{v}_i\rangle_T$, where m is the number of components ($\mathbf{u}_i, i = 1,\dots,m$) of $\mathbf{u}$. Moreover, we define $(\mathbf{u},\mathbf{v})_{\mathcal{T}_h} := \sum_{T\in\mathcal{T}_h} (\mathbf{u},\mathbf{v})_T$ and $\langle\mathbf{u},\mathbf{v}\rangle_{\mathcal{E}_h} := \sum_{e\in\mathcal{E}_h} \langle\mathbf{u},\mathbf{v}\rangle_e$ whose induced norms are clear, and hence their definitions are omitted. We employ boldface uppercase letters, e.g., $\mathbf{K}$, to denote matrices and tensors. We denote by $\mathcal{P}^p(T)$ the space of polynomials of degree at most p on a domain T. We use the terms "skeletal unknowns," "trace unknowns," and "Lagrange multipliers" interchangeably and they all refer to the unknowns on the mesh skeleton.

First, we cast (2.1) into the following first-order or mixed form:

$$\mathbf{u} = -\mathbf{K}\nabla q \quad \text{in } \Omega, \tag{2.2a}$$

$$\nabla\cdot\mathbf{u} = f \quad \text{in } \Omega, \tag{2.2b}$$

$$q = g_D \quad \text{on } \partial\Omega. \tag{2.2c}$$

The hybrid mixed DG method or hybridized DG (HDG) method for the discretization of (2.2) is defined as

$$\left(\mathbf{K}^{-1}\mathbf{u},\mathbf{v}\right)_T - (q,\nabla\cdot\mathbf{v})_T + \langle\lambda,\mathbf{v}\cdot\mathbf{n}\rangle_{\partial T} = 0, \tag{2.3a}$$

$$-(\mathbf{u},\nabla w)_T + \langle\hat{\mathbf{u}}\cdot\mathbf{n}, w\rangle_{\partial T} = (f,w)_T, \tag{2.3b}$$

$$\langle[\![\hat{\mathbf{u}}\cdot\mathbf{n}]\!],\mu\rangle_e = 0, \tag{2.3c}$$

where the numerical flux $\hat{\mathbf{u}}\cdot\mathbf{n}$ is given by

$$\hat{\mathbf{u}}\cdot\mathbf{n} = \mathbf{u}\cdot\mathbf{n} + \tau(q-\lambda). \tag{2.4}$$

For simplicity, we have ignored the fact that (2.3a), (2.3b), and (2.3c) must hold for all test functions $\mathbf{v}\in\mathbf{V}_h(T)$, $w\in W_h(T)$, and $\mu\in M_h(e)$, respectively (this is implicitly understood throughout the paper), where $\mathbf{V}_h$, W_h, and M_h are defined as

$$\mathbf{V}_h(\mathcal{T}_h) = \left\{\mathbf{v}\in\left[L^2(\mathcal{T}_h)\right]^d : \mathbf{v}|_T \in [\mathcal{P}^p(T)]^d\, \forall T\in\mathcal{T}_h\right\},$$
$$W_h(\mathcal{T}_h) = \left\{w\in L^2(\mathcal{T}_h) : w|_T\in\mathcal{P}^p(T)\,\forall T\in\mathcal{T}_h\right\},$$
$$M_h(\mathcal{E}_h) = \left\{\lambda\in L^2(\mathcal{E}_h) : \lambda|_e\in\mathcal{P}^p(e)\,\forall e\in\mathcal{E}_h\right\},$$

and similar spaces $\mathbf{V}_h(T)$, $W_h(T)$, and $M_h(e)$ on T and e can be defined by replacing $\mathcal{T}_h$ with T and $\mathcal{E}_h$ with e, respectively.

Next, we consider the hybridized interior penalty DG (IPDG) schemes posed in the primal form. To that end, we test (2.2a) with $\mathbf{v} = \nabla w$ and then integrate by parts twice the terms on the right-hand side to obtain

$$(\mathbf{u},\nabla w)_T = -(\mathbf{K}\nabla q,\nabla w)_T + \langle(q-\lambda),\mathbf{K}\nabla w\cdot\mathbf{n}\rangle_{\partial T}.$$

Substituting this in (2.3b) gives

$$(\mathbf{K}\nabla q, \nabla w)_T - \langle (q-\lambda), \mathbf{K}\nabla w\cdot\mathbf{n}\rangle_{\partial T} + \langle \hat{\mathbf{u}}\cdot\mathbf{n}, w\rangle_{\partial T} = (f,w)_T\,. \tag{2.5}$$

For IPDG schemes, the numerical flux takes the form

$$\hat{\mathbf{u}}\cdot\mathbf{n} = -\mathbf{K}\nabla q\cdot\mathbf{n} + \tau(q-\lambda), \tag{2.6}$$

and it is required to satisfy the conservation condition (2.3c). Substituting the numerical flux (2.6) in (2.5) we get the following primal form of the hybridized IPDG scheme:

$$\begin{aligned}(\mathbf{K}\nabla q, \nabla w)_T - \langle (q-\lambda), \mathbf{K}\nabla w\cdot\mathbf{n}\rangle_{\partial T} &- \langle \mathbf{K}\nabla q\cdot\mathbf{n}, w\rangle_{\partial T}\\ &+ \langle \tau(q-\lambda), w\rangle_{\partial T} = (f,w)_T\,,\end{aligned} \tag{2.7}$$

This scheme is called the hybridized symmetric IPDG scheme (SIPG-H) and has been studied in [20, 28, 45]. In order to include the other hybridized IPDG schemes, we can generalize (2.7) to

$$\begin{aligned}(\mathbf{K}\nabla q, \nabla w)_T - s_f\,\langle (q-\lambda), \mathbf{K}\nabla w\cdot\mathbf{n}\rangle_{\partial T} &- \langle \mathbf{K}\nabla q\cdot\mathbf{n}, w\rangle_{\partial T}\\ &+ \langle \tau(q-\lambda), w\rangle_{\partial T} = (f,w)_T\,,\end{aligned} \tag{2.8}$$

where $s_f \in \{-1,0,1\}$. The scheme corresponding to $s_f = -1$ is called the hybridized nonsymmetric IPDG scheme (NIPG-H) and the one with $s_f = 0$ is called the hybridized incomplete IPDG scheme (IIPG-H) [28]. It is interesting to note that in the HDG scheme, if we do not integrate by parts (2.3a) and substitute $\mathbf{u} = -\mathbf{K}\nabla q$ in (2.3b) and (2.4) we obtain the IIPG-H scheme. On the other hand, if we do not integrate by parts (2.3b) and take $\hat{\mathbf{u}}\cdot\mathbf{n} = \mathbf{u}\cdot\mathbf{n}$ in (2.3c) we obtain the hybridized mixed methods

$$\left(\mathbf{K}^{-1}\mathbf{u}, \mathbf{v}\right)_T - (q, \nabla\cdot\mathbf{v})_T + \langle \lambda, \mathbf{v}\cdot\mathbf{n}\rangle_{\partial T} = 0, \tag{2.9a}$$

$$(\nabla\cdot\mathbf{u}, w)_T = (f,w)_T\,, \tag{2.9b}$$

and we need to enforce the conservation condition $\langle [\![\mathbf{u}\cdot\mathbf{n}]\!], \mu\rangle_e = 0$. However, in this case we can no longer choose the same solution order for both $\mathbf{u}$ and q due to the lack of stabilization. Indeed, inf-sup stable mixed finite element spaces have to be chosen for $\mathbf{V}_h$ and W_h. For the details, we refer readers to [19] for the hybridized Raviart–Thomas (RT-H) and the hybridized Brezzi–Douglas–Marini (BDM-H) mixed finite element spaces.

We note in passing that the hybridized weak Galerkin mixed finite element methods introduced in [33] follow similar hybridization as in the hybridized mixed methods but with a different choice for local spaces accompanied with a stabilization term. In many cases one can show that HDG and weak Galerkin methods coincide, and we do not attempt to distinguish them in this paper.

The common solution procedure for all of these hybridized methods can now be described. First, we express the local volume unknowns $\mathbf{u}$ and/or q, element by element, as a function of the skeletal unknowns λ. Then, we use the conservation condition to construct a global linear system involving only the skeletal unknowns. Once they are solved for, the local volume unknowns can be recovered in an element-by-element fashion completely independent of each other. The main advantage of this Schur complement approach is that, for high-order methods, the global trace system is much smaller and sparser compared to the linear system for the volume unknowns [13, 20]. The question that needs to be addressed is how to solve the trace system efficiently. In the next section we develop a geometric multigrid algorithm to provide an answer.

3. Geometric multigrid algorithm based on DtN maps.

For all of the hybridized methods described in the previous section, the resulting linear system for the skeletal unknowns λ, in operator form, can be written as

$$A\lambda = g. \tag{3.1}$$

The well-posedness of the trace system (3.1) for the hybridized methods discussed in the previous section has been shown in [20, 28, 33].

The concept of a DtN map is essential to our approach, so we briefly describe it here. A map A is called a DtN map/operator if it maps Dirichlet data on a domain boundary to Neumann data. It is a particular type of Poincaré–Steklov operator, which contains a large class of operators which map one type of boundary condition to another for elliptic PDEs. In finite dimension, the Schur complement of the linear system with volume unknowns condensed out is also known as a discrete DtN map. We refer readers to [35] for more information.

To define our multigrid algorithm we start with a sequence of partitions of the mesh $\mathcal{T}_h$:

$$\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N = \mathcal{T}_h,$$

where each $\mathcal{T}_k$ contains N_{T_k} closed (not necessarily convex, except on the finest level where each N_{T_k} is in fact some element T_j in $\mathcal{T}_h$) macro-elements consisting of unions of macro-elements from $\mathcal{T}_{k+1}$. Associated with each partition, we define interface grids $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_N = \mathcal{E}_h$, where each $e \in \mathcal{E}_k$ is the intersection of two macro-elements in $\mathcal{T}_k$. Each partition $\mathcal{E}_k$ is in turn associated with a skeletal (trace) space M_k (to be defined in section 3.1).

Remark 3.1. We note that if each $\mathcal{T}_k$ consists of simplices or parallelopipeds, then each M_k is straightforward to define. In general, however, the macro-elements need not be one of these standard shapes, and we need to define a parameterization of each macro-edge to define M_k. While this is certainly nontrivial, this calculation only needs to be performed one time for each mesh and can be reused for many simulation or time steps as long as the macro-elements (agglomeration) do not change. For all of the results presented in section 4, the time required to calculate these parameterizations is negligible in comparison with the total simulation time.

We are now in position to introduce necessary ingredients to define our geometric multigrid algorithm.

3.1. Coarsening strategy.

In the multigrid algorithm we first coarsen in p, followed by the coarsening in h. We explain these two coarsening procedures in detail as follows.

3.1.1. *p*-coarsening.

In our p-coarsening strategy, we restrict the solution order p on the finest level N to $p = 1$ on level $N-1$ with the same number of elements. In this case the Lagrange multiplier spaces become

$$M_k = \begin{cases} \{\eta \in \mathcal{P}^p(e) \forall e \in \mathcal{E}_k\} & \text{for} \quad k = N, \\ \{\eta \in \mathcal{P}^1(e) \forall e \in \mathcal{E}_k\} & \text{for} \quad k = 1, 2, \ldots, N-1. \end{cases}$$

Let us comment on one subtle point. It would be ideal if we could use $\mathcal{P}^0$ in coarser levels because it is trivial to define piecewise constant functions along arbitrary macro-edges (two-dimensional) or macro-faces (three-dimensional), i.e., no parameterization is needed. However, multilevel algorithms using restriction and prolongation operators based on piecewise constant interpolation typically have been shown not to perform well [4, 32].

For high-order methods, the numerical examples in section 4 indicate that our strategy gives scalable results in solution order p when strong smoothers such as block-Jacobi or Gauss–Seidel are used. This is also observed in [30] for a p-multigrid approach applied to DG methods for the Poisson equation using block-Jacobi smoother. Other p-multigrid strategies, such as $p_{k-1} = p_k/2$ (p_k is the solution order on the kth level) or $p_{k-1} = p_k - 1$, can be straightforwardly incorporated within our approach. Our future work will include theoretical and numerical comparisons among these strategies. Once we interpolate to $p = 1$ we carry out the h-coarsening as described in the next section.

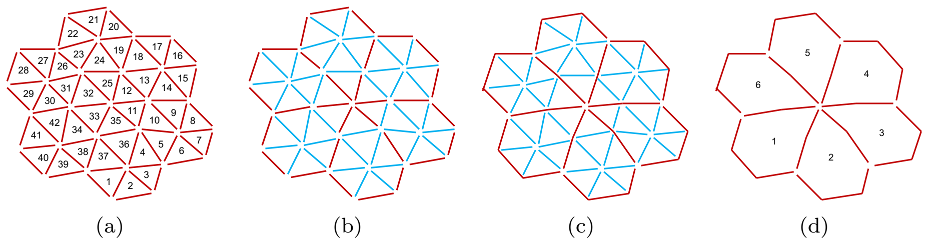


Fig. 3.1. *A demonstration of h-coarsening strategy.* (a) *Level k mesh.* (b) *An identification of interior $\mathcal{E}_{k,I}$ (blue) and boundary $\mathcal{E}_{k,B}$ (red) edges.* (c) *Coarsening of boundary edges.* (d) *Level $k-1$ mesh after interior edges are statically condensed out. The numbers in* (a) *and* (d) *represent the number of (macro-) elements in levels k and $k-1$, respectively.*

3.1.2. *h*-coarsening. We adopt an agglomeration-based h-coarsening, allowing intergrid transfer and coarse grid operators to be defined using the fine scale DtN maps. By taking this approach, no upscaling of parameters is required and our coarse operators are discretized DtN maps at any mesh level (see Proposition 3.3). In Figure 3.1, we show an h-coarsening strategy between two consecutive levels k and $k-1$. First, we identify the interior (blue) and boundary (red) edges in Figure 3.1(b). We decompose $\mathcal{E}_k = \mathcal{E}_{k,I} \oplus \mathcal{E}_{k,B}$, where $\mathcal{E}_{k,I}$ consists of edges interior to macro-elements in the coarser partition $\mathcal{T}_{k-1}$ and $\mathcal{E}_{k,B}$ contains edges common to their boundaries. We also decompose the trace space M_k on $\mathcal{E}_k$ into two parts $M_{k,I}$ and $M_{k,B}$ corresponding to $\mathcal{E}_{k,I}$ and $\mathcal{E}_{k,B}$, respectively. Specifically, we require $M_k = M_{k,I} \oplus M_{k,B}$ such that each $\lambda_k \in M_k$ can be uniquely expressed as $\lambda_k = \lambda_{k,I} + \lambda_{k,B}$, where

$$\lambda_{k,I} = \begin{cases} \lambda_k & \text{on } M_{k,I}, \\ 0 & \text{on } M_{k,B}, \end{cases} \quad \text{and} \quad \lambda_{k,B} = \begin{cases} 0 & \text{on } M_{k,I}, \\ \lambda_k & \text{on } M_{k,B}. \end{cases}$$

Given the decomposition $M_k = M_{k,I} \oplus M_{k,B}$, the trace system (3.1) at the kth level can be written as

$$A_k \lambda_k = g_k \Leftrightarrow \begin{bmatrix} A_{k,II} & A_{k,IB} \\ A_{k,BI} & A_{k,BB} \end{bmatrix} \begin{bmatrix} \lambda_{k,I} \\ \lambda_{k,B} \end{bmatrix} = \begin{bmatrix} g_{k,I} \\ g_{k,B} \end{bmatrix}. \tag{3.2}$$

The coarser space M_{k-1} is defined such that $M_{k-1} \subset M_{k,B}$. This is done by first agglomerating the boundary edges of level k as in Figure 3.1(c) and then statically condensing out the interior edges to obtain the mesh on level $k-1$ in Figure 3.1(d). The elements in level k are numbered from 1 to 42 in Figure 3.1(a) and the macro elements in level $k-1$ are numbered from 1 to 6 in Figure 3.1(d).

Clearly, for an unstructured mesh the identification of interior and boundary edges is nontrivial and nonunique. In this paper we use an ad hoc approach which is sufficient for the purpose of demonstrating our proposed algorithm. Specifically, we first select a certain number of levels N and seed points N_{T_1}. The locations of these seed points are chosen based on the geometry of the domain and the original fine mesh, such that we approximately have an equal number of fine mesh elements in each macro-element at level 1. Based on these selected seed points, we agglomerate the elements in the finest mesh ($k = N$) to form N_{T_1} macro-elements at level 1. We then divide each macro-element in level 1 into four approximately equal macro-elements to form level 2. This process is recursively repeated to create macro-elements in finer levels $k = 3, \ldots, N-1$. As an illustration we consider the mesh in Figure 3.2 (this mesh and the corresponding coarsening strategies will be used in numerical example II in section 4.2) and in Figures 3.3 and 3.4 we show two different coarsening strategies with $N = 7$ obtained from seven and four seed points.

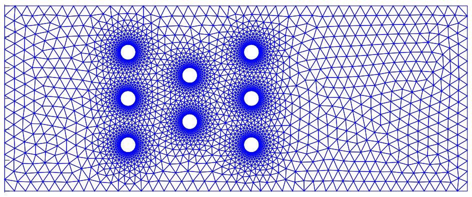


Fig. 3.2. *Example* II: *Unstructured mesh for a rectangular box with eight holes.*

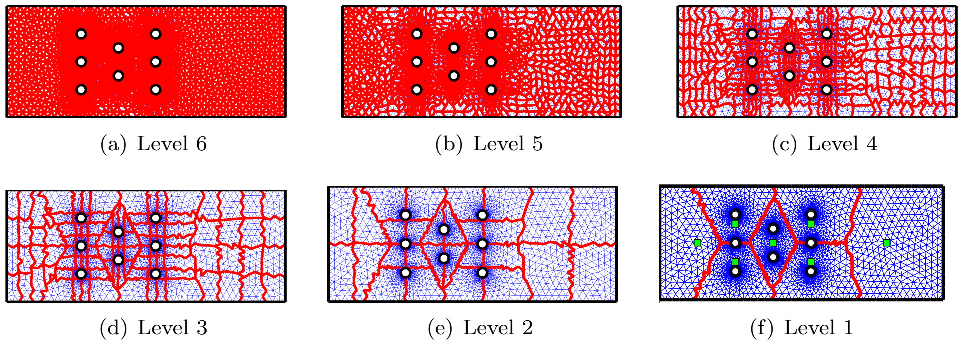


Fig. 3.3. *Example* II. *Coarsening strategy* 1: *the seed points are marked with green squares in level* 1.

Part of our future work is to explore the coarsening strategies similar to the ones in [42, 43]. This may give better coarsened levels with the number of macro-elements reduced/increased by a constant factor between successive levels. The intergrid transfer operators necessary for moving residuals/errors between levels is described next.

3.2. Intergrid transfer operators. In standard nested multigrid algorithms, the intergrid transfer operators are straightforward. A popular approach is to take

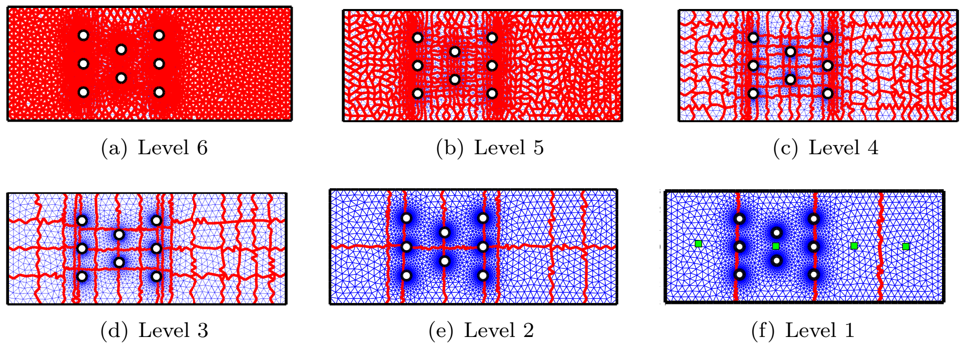

(a) Level 6 (b) Level 5 (c) Level 4 (d) Level 3 (e) Level 2 (f) Level 1

FIG. 3.4. *Example* II. *Coarsening strategy* 2: *the seed points are marked with green squares in level* 1.

injection for the prolongation and its adjoint for the restriction. For a skeletal system care must be taken in the construction of these operators to ensure the convergence of multigrid algorithm under consideration. In the following we propose "physics-based energy-preserving" operators using the fine scale DtN maps.

3.2.1. Prolongation. The prolongation operator, $I_k : M_{k-1} \longrightarrow M_k$, transfers the error from the grid level $k-1$ to the finer grid level k. Note that standard prolongation using injection would set the values on the interior edges to zero and thus does not work. For our multigrid algorithm it is defined as a function of the solution order p using DtN maps. In particular,

- for $p = 1$,

$$I_k := \begin{bmatrix} -A_{k,II}^{-1} A_{k,IB} J_k \\ J_k \end{bmatrix} \quad \text{for} \quad k = 2, \dots, N, \tag{3.3}$$

- and for $p > 1$,

$$I_k := \begin{cases} J_N & \text{for} \quad k = N, \\ \begin{bmatrix} -A_{k,II}^{-1} A_{k,IB} J_k \\ J_k \end{bmatrix} & \text{for} \quad k = 2, \dots, N-1. \end{cases} \tag{3.4}$$

Here, $J_k : M_{k-1} \longrightarrow M_k$ is the injection (interpolation). To understand the idea behind the prolongation operator, let us consider $p = 1$ in (3.3). First, through J_k, we interpolate the error from the grid level $k-1$ to obtain error on the boundary edges of the finer grid level k. Then we solve (via the first block in the definition of I_k) for the error on the interior edges as a function of the interpolated error on the boundary. For $p > 1$ the prolongation is defined in (3.4), namely, we use the same prolongation as in the case of $p = 1$ for all levels $k \le N-1$ and interpolate error from piecewise linear polynomials at level $N-1$ to piecewise pth-order polynomials at level N.

3.2.2. Restriction. The restriction operator, $Q_{k-1} : M_k \longrightarrow M_{k-1}$, restricts the residual from level k to coarser level $k-1$. The popular idea is to define the restriction operator as the adjoint (with respect to the L^2-inner products $\langle\cdot,\cdot\rangle_{\mathcal{E}_k}$ and

$\langle \cdot, \cdot \rangle_{\mathcal{E}_{k-1}}$ in M_k and M_{k-1}) of the prolongation operator, i.e., $Q_{k-1} = I_k^*$, and from our numerical studies (not shown here) it still works well. However, this purely algebraic procedure, though convenient, is not our desire. Here, we construct the restriction operator Q_{k-1} such that the coarse grid problem, via the Galerkin approximation, is exactly a discretized DtN problem on level $k-1$. To that end, let us define Q_{k-1} as

- for $p = 1$,

$$Q_{k-1} := \begin{bmatrix} -J_k^* A_{k,BI} A_{k,II}^{-1} & J_k^* \end{bmatrix} \quad \text{for} \quad k = 2, \dots, N, \tag{3.5}$$

- for $p > 1$,

$$Q_{k-1} := \begin{cases} J_N^* & \text{for} \quad k = N, \\ \begin{bmatrix} -J_k^* A_{k,BI} A_{k,II}^{-1} & J_k^* \end{bmatrix} & \text{for} \quad k = 2, \dots, N-1. \end{cases} \tag{3.6}$$

Note that if A is symmetric, then our definition of the restriction operator Q_{k-1} is indeed the adjoint of the prolongation operator I_k.

Given the prolongation and restriction operators, we obtain our coarse grid equation using the discrete Galerkin approximation [41]

$$Q_{k-1} A_k I_k \lambda_{k-1} = Q_{k-1} g_k, \tag{3.7}$$

where the coarse grid operator

$$A_{k-1} := Q_{k-1} A_k I_k, \tag{3.8}$$

for either $p = 1$ and $k \leq N$ or $p > 1$ and $k \leq N - 1$, reads

$$\begin{aligned} A_{k-1} &= \begin{bmatrix} -J_k^* A_{k,BI} A_{k,II}^{-1} & J_k^* \end{bmatrix} \begin{bmatrix} A_{k,II} & A_{k,IB} \\ A_{k,BI} & A_{k,BB} \end{bmatrix} \begin{bmatrix} -A_{k,II}^{-1} A_{k,IB} J_k \\ J_k \end{bmatrix} \\ &= J_k^* \left(A_{k,BB} - A_{k,BI} A_{k,II}^{-1} A_{k,IB} \right) J_k, \end{aligned} \tag{3.9}$$

and

$$A_{N-1} = J_N^* A_N J_N, \tag{3.10}$$

for $p > 1$ and $k = N$.

Energy preservation. In order for the multigrid algorithms to converge the intergrid operators should be constructed in such a way that the "energy" does not increase when transferring information between a level to a finer one [5, 29]. Note that "energy" here need not necessarily be associated with some physical energy. Indeed, if we associate A_k with a bilinear form $a_k(.,.)$ such that $a_k(\kappa, \mu) = \langle A_k \kappa, \mu \rangle_{\mathcal{E}_k} \, \forall \kappa, \mu \in M_k$, where again $\langle ., . \rangle_{\mathcal{E}_k}$ represents the L^2-inner product on $\mathcal{E}_k$, then we call $a_k(\kappa, \kappa)$ the energy on level k associated with κ. Nonincreasing energy means

$$a_k(I_k \lambda, I_k \lambda) \leq a_{k-1}(\lambda, \lambda) \quad \forall \lambda \in M_{k-1}, \quad \forall k = 2, 3, \cdots, N.$$

PROPOSITION 3.2 (energy preservation). *The proposed multigrid approach preserves the energy in the following sense:* $\forall k = 2, 3, \dots, N$,

$$a_k(I_k \lambda, I_k \lambda) = a_{k-1}(\lambda, \lambda) \quad \forall \lambda \in M_{k-1}. \tag{3.11}$$

Proof. We proceed first with $p = 1$. From the definition of A_k in (3.2) and the definition of the prolongation operator I_k in (3.3) we have

$$\begin{aligned} & a_k\left(I_k\lambda, I_k\lambda\right) \\ &= \left\langle \left[\begin{array}{cc} -A_{k,II}^{-1}A_{k,IB}J_k\lambda, J_k\lambda \end{array}\right], \left[\begin{array}{cc} A_{k,II} & A_{k,IB} \\ A_{k,BI} & A_{k,BB} \end{array}\right] \left[\begin{array}{c} -A_{k,II}^{-1}A_{k,IB}J_k\lambda \\ J_k\lambda \end{array}\right] \right\rangle_{\mathcal{E}_k} \\ &= \left\langle \left(A_{k,BB} - A_{k,BI}A_{k,II}^{-1}A_{k,IB}\right) J_k\lambda, J_k\lambda \right\rangle_{\mathcal{E}_k} \\ &= \left\langle J_k^*\left(A_{k,BB} - A_{k,BI}A_{k,II}^{-1}A_{k,IB}\right) J_k\lambda, \lambda \right\rangle_{\mathcal{E}_{k-1}} = a_{k-1}\left(\lambda,\lambda\right), \end{aligned}$$

where the last equality comes from the definition of the coarse grid operator A_{k-1} in (3.9). For $p > 1$, we need to prove (3.11) only for $k = N$, but this is straightforward since from (3.4), (3.6), and (3.10) we have

$$a_N\left(I_N\lambda, I_N\lambda\right) = \left\langle A_N J_N\lambda, J_N\lambda\right\rangle_{\mathcal{E}_k} = \left\langle J_N^* A_N J_N\lambda, \lambda\right\rangle_{\mathcal{E}_{k-1}} = a_{k-1}\left(\lambda,\lambda\right). \qquad \square$$

We now prove that the coarse grid operator (3.8) is also a discretized DtN map on every level.

PROPOSITION 3.3. *At every level* $k = 1, \ldots, N$, *the Galerkin coarse grid operator* (3.8) *is a discretized DtN map on that level.*

Proof. The proof is a straightforward induction using the proposed coarsening strategy and the definition of the intergrid transfer operators. First, consider the case of $p = 1$. The fact that $A_N = A$ in (3.1) is a discretized DtN map on the finest level is clear by the definition of the hybridized methods. Assume at level k the operator A_k in (3.2) is a discretized DtN map. Taking $\lambda_{k,B} = J_k\lambda_{k-1}$ and condensing $\lambda_{k,I}$ out yield

$$\left(A_{k,BB} - A_{k,BI}A_{k,II}^{-1}A_{k,IB}\right) J_k\lambda_{k-1} = g_{k,B} - A_{k,BI}A_{k,II}^{-1}g_{k,I}$$

which, after restricting on the coarse space M_{k-1} using J_k^*, becomes

$$J_k^*\left(A_{k,BB} - A_{k,BI}A_{k,II}^{-1}A_{k,IB}\right) J_k\lambda_{k-1} = J_k^*\left(g_{k,B} - A_{k,BI}A_{k,II}^{-1}g_{k,I}\right),$$

which is exactly the coarse grid equation (3.7). That is, A_{k-1} is the Schur complement obtained by eliminating all the trace unknowns inside all the macro-elements on level $k-1$. By definition, the coarse grid operator A_{k-1} on level $(k-1)$ is also a discrete DtN map.

For the case of $p > 1$, it is sufficient to show that A_{N-1} is a discrete DtN map, but this is clear by: (1) taking $\lambda = J_N\lambda_{N-1}$ in (3.1), (2) restricting both sides to M_{N-1} using J_N^*, (3) recalling that A_N is a discretized DtN map, and (4) observing that the resulting equation coincides with the coarse grid equation (3.10). $\square$

3.3. Smoothing. Let us denote the smoothing operator at level k by G_{k,m_k}, where m_k stands for the number of smoothing steps performed at level k. We take m_k to satisfy

$$\beta_0 m_k \le m_{k-1} \le \beta_1 m_k,$$

with $\beta_1 \ge \beta_0 > 1$. That is, the number of smoothing steps is increased as the mesh is coarsened. The reason for this choice is based on the theoretical analysis for nonnested multigrid methods in [6]. However, as numerically shown later in section 4.5, even a constant number of smoothing steps at all levels, e.g., two, works well.

3.4. Local correction operator. To motivate the need for a local correction operator, let us consider the following decomposition of the skeletal space $M_k = \overline{M}_k \oplus \hat{M}_k$ such that

$$\lambda_k = \overline{\lambda}_k + \hat{\lambda}_k, \quad \overline{\lambda}_k \in \overline{M}_k \text{ and } \hat{\lambda}_k \in \hat{M}_k,$$

where $\overline{\lambda}_k = \overline{\lambda}_{k,I}$ is given by the local correction $T_k : M_k \longrightarrow M_k$,

$$\overline{\lambda}_k := T_k \begin{bmatrix} g_{k,I} \\ 0 \end{bmatrix} := \begin{bmatrix} A_{k,II}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} g_{k,I} \\ 0 \end{bmatrix}, \tag{3.12}$$

and the "global component" $\hat{\lambda}_k = \begin{bmatrix} \hat{\lambda}_{k,I} & \hat{\lambda}_{k,B} \end{bmatrix}^T$ by

$$\begin{bmatrix} A_{k,II} & A_{k,IB} \\ A_{k,BI} & A_{k,BB} \end{bmatrix} \begin{bmatrix} \hat{\lambda}_{k,I} \\ \hat{\lambda}_{k,B} \end{bmatrix} = \begin{bmatrix} 0 \\ g_{k,B} - A_{k,BI} A_{k,II}^{-1} g_{k,I} \end{bmatrix}.$$

Let us now consider $p = 1$. Standard two-grid cycles, for example, include (1) smoothing iterations on the system (3.1) for the fine grid, (2) restricting the residual to the coarse grid, (3) performing the coarse grid correction, and (4) prolongating the error back to the fine grid. However, with the prolongation operator defined in (3.3), only $\hat{\lambda}_k$ can be recovered directly. In other words, the burden of capturing $\overline{\lambda}_k$ is left for the smoother and/or multigrid iterations. Indeed, in our numerical studies we found that this standard algorithm takes more multigrid iterations and still converges with Gauss–Seidel and Chebyshev accelerated Jacobi smoothers. However, it diverges with the block-Jacobi smoother at low orders as shown in Table 4.8.

This issue can be fixed using the concept of subspace correction [41, 49]. We can perform a local correction in the subspace $\overline{M}_k$, as in (3.12), either before or after the coarse grid correction (or both if symmetry is desirable). With this local correction, we have observed that even with Jacobi-type smoothers the multigrid algorithm converges. It is important to point out that the application of the local correction operator T_k is completely local to each macro-element at the $(k-1)$th level and thus can be carried out in parallel.

3.5. Relationship to algebraic multigrid operators. The intergrid transfer operators in section 3.2 and the local correction operator in section 3.4 are closely related to the ideal interpolation and smoothing operators in the algebraic multigrid (AMG) literature [41]. However, the ideal operators in AMG lead to a direct solver strategy, namely, nested dissection [26], and suffer from large memory requirements. This renders the algorithm impractical for large scale simulations especially in three dimensions. In that respect, the coarsening in our operators is the key as it leads to an $\mathcal{O}(N)$ iterative algorithm and can be applied to large scale problems. In fact, this approach is pursued under the name of Schur complement multigrid methods in [44] and similar works [22, 23, 36, 37], mostly in the context of finite differences and finite volumes. Here we apply similar ideas for high-order hybridized finite element methods and multinumerics. Also our approach is more general in the sense that it can be applied to any unstructured mesh, whereas all the previous works deal with structured meshes. As shown in Figure 3.2, it does not require the meshes to be nested. In fact, our approach can be considered as a combination of AMG and geometric multigrid methods. As such it benefits from the robustness of AMG and the fixed coarse grid construction costs of geometric multigrid.

3.6. Multigrid V-cycle algorithm. We are now in position to define our multigrid algorithm. We begin with a fixed point scheme on the finest level:

$$\lambda^{i+1} = \lambda^i + B_N(r_N), \quad i = 0, \ldots .$$

Algorithm 3.1 Multigrid v-cycle algorithm

```
1: Initialization:
   $e^{\{0\}} = 0,$
2: Presmoothing:
   $e^{\{1\}} = e^{\{0\}} + G_{k,m_k}\left(r_k - A_k e^{\{0\}}\right),$
3: Local correction:
   $e^{\{2\}} = e^{\{1\}} + T_k\left(r_k - A_k e^{\{1\}}\right),$
4: Coarse grid correction:
   $e^{\{3\}} = e^{\{2\}} + I_k B_{k-1}\left(Q_{k-1}\left(r_k - A_k e^{\{2\}}\right)\right),$
5: Local correction:
   $e^{\{4\}} = e^{\{3\}} + T_k\left(r_k - A_k e^{\{3\}}\right),$
6: Postsmoothing:
   $B_k\left(r_k\right) = e^{\{5\}} = e^{\{4\}} + G_{k,m_k}\left(r_k - A_k e^{\{4\}}\right).$
```

At the coarsest level M_1, we set $B_1 = A_1^{-1}$ and the inversion is computed using a direct solver. Note that both local correction steps 3 and 5 are presented for the sake of symmetry of the algorithm. In practice, we use either step 3 or step 5. For example, numerical results in section 4 use only step 3.

4. Numerical results.

In this section, we study the performance of multigrid both as a solver and as a left preconditioner. Since the global trace system of the NIPG-H and IIPG-H methods is not symmetric, we use preconditioned GMRES for all the methods to enable a direct comparison. For all the numerical examples, (1) the stopping tolerance is taken as 10^{-9} for the normalized residual (normalized by the norm of the right-hand side of (3.1)), and (2) following [6] we choose the smoothing parameters β_0 and β_1 (see section 3.3) as 2. In all the tables, "$*$" stands for either the divergence of the GMRES/multigrid solver or 200 iterations were already taken but the normalized residual was still larger than the tolerance.

In examples I–V in sections 4.1–4.5, we consider HDG and hybridized IPDG methods. In example VI in section 4.6, we consider the case of multinumerics with mixed methods (RT-H) in some parts of the domain and NIPG-H in other parts. Though we have also tested the multigrid algorithm for hybridized RT1 and RT0 methods for example I and obtained scalable results, they are not shown here due to the page limitation.

4.1. Example I: Poisson equation in the unit square.

In this first example, we consider the Poisson equation in the unit square. We take the exact solution to be of the form $q = xye^{x^2y^3}$. Dirichlet boundary conditions using the exact solution are applied directly (strongly) via the trace unknowns. We consider HDG and hybridized IPDG methods and study the performance of multigrid both as a solver and as a preconditioner. The number of levels in the multigrid hierarchy and the corresponding number of quadrilateral elements are shown in Table 4.1.

TABLE 4.1
Example I: *The multigrid hierarchy.*

Levels	2	3	4	5	6	7
Elements	16	64	256	1024	4096	16384

As a representative study of point and block smoothers we consider the following two smoothers and compare their performance for the hybridized DG methods.

- Lower-upper symmetric point Gauss–Seidel method with one forward and one backward sweep during each iteration; we call this smoother LU-SGS for simplicity.
- Block–Jacobi smoother where one block consists of all the degrees of freedom corresponding to an edge $e \in \mathcal{E}_k$ at the kth level. We do not use any damping as it does not make any difference in the number of iterations in our numerical studies.

We refer readers to our report in [48] for a more comprehensive presentation, where we also show results for point Jacobi and Chebyshev accelerated point Jacobi smoothers. Similarly in [48] we present results for solution orders from 1 to 10, whereas here we show only representative results for the sake of brevity.

For the stabilization parameters, we consider a mesh-independent value $\tau = 1$ and a mesh-dependent form $\tau = 1/h_{min}$ for HDG. The former corresponds to the upwind HDG proposed in [11, 12]. Following [38], we take the stabilization to be $\tau = (p+1)(p+2)/h_{min}$ for the hybridized IPDG schemes. In later examples, when the permeability $\mathbf{K}$ is spatially varying we also consider τ as a function of $\mathbf{K}$. We refer to [11, 17, 21] for the h-convergence order of HDG schemes and [38] for the IPDG schemes.

In Tables 4.2 and 4.3, we study the performance of multigrid as a solver and as a preconditioner for HDG with stabilization $\tau = 1/h_{min}$. We now present a few observations from these tables. The stabilization $\tau = 1/h_{min}$ gives h-scalable results with both the smoothers, i.e., the number of required multigrid/GMRES iterations is almost unchanged when the mesh is refined. Table 4.2 shows that LU-SGS is a better smoother than block-Jacobi smoother in Table 4.3, in terms of multigrid/GMRES iterations. Moreover in Table 4.2, the numbers of both multigrid and GMRES iterations are almost constant as either the mesh is refined or the solution order increases. However, it should be pointed out that the LU-SGS smoother requires twice the amount of work compared to point-Jacobi for each iteration due to one forward and one back-

Table 4.2
Example I. *HDG with stabilization $\tau = 1/h_{min}$: the number of iterations for multigrid with LU-SGS smoother as solver and preconditioner.*

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	5	5	5	5	5	5	4	4	4	5	5	5
5	7	7	7	7	7	7	5	5	6	6	6	5
10	9	9	10	10	10	10	7	7	7	7	7	7

Table 4.3
Example I. *HDG with stabilization $\tau = 1/h_{min}$: the number of iterations for multigrid with block-Jacobi smoother as solver and preconditioner.*

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	7	7	8	8	8	8	4	5	6	6	6	6
2	6	7	8	8	9	9	4	5	6	6	6	6
3	8	9	9	9	9	9	6	6	6	6	6	6
4	9	10	10	10	10	10	6	7	7	7	7	7
7	13	14	14	14	14	15	7	8	8	8	8	8
10	16	17	17	17	17	17	8	9	9	9	9	9

ward sweep. Compared to the point-Jacobi smoother in [48], Table 4.3 shows that the block-Jacobi smoother requires fewer multigrid/GMRES iterations; otherwise its scalability behavior is similar to the point-Jacobi smoother in [48] as the mesh or the solution order is refined.

In Tables 4.4 and 4.5, we present results for the multigrid solver and GMRES with multigrid preconditioner when $\tau = 1$ is used for the HDG discretization. With the LU-SGS smoother (Table 4.4), the number of iterations for the multigrid solver scales like $\mathcal{O}(1/h)$ (since the number of iterations is nearly doubled each time h reduces by half). Even for GMRES with multigrid as preconditioner, we do not obtain h-scalability. On the other hand, the block-Jacobi smoother in Table 4.5 for $p \geq 4$ provides h-scalable results and with solution orders greater than 5 the results are exactly the same as those for $\tau = 1/h_{min}$. With respect to solution orders, similar to the case of $\tau = 1/h_{min}$, we do not obtain perfect p-scalability with both the smoothers though the growth of the number of iterations is very slow with the block-Jacobi smoother.

We present in Tables 4.6 and 4.7 the results for the hybridized NIPG-H scheme with $\tau = (p + 1)(p + 2)/h_{min}$. As can be seen, the number of iterations for multigrid both as a solver and as a preconditioner is similar to those with HDG using $\tau = 1/h_{min}$. The results using IIPG-H and SIPG-H are similar and hence are omitted for brevity.

TABLE 4.4
Example I. *HDG with stabilization $\tau = 1$: the number of iterations for multigrid with LU-SGS smoother as solver and preconditioner.*

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	6	11	21	40	76	145	4	7	11	15	21	28
5	13	21	36	66	124	*	8	10	14	18	24	32
10	17	26	43	75	137	*	8	11	14	18	24	31

TABLE 4.5
Example I. *HDG with stabilization $\tau = 1$: the number of iterations for multigrid with block-Jacobi smoother as solver and preconditioner.*

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	5	8	15	27	51	97	4	6	9	12	17	23
2	5	7	9	14	24	42	4	5	7	9	12	15
3	8	9	9	9	15	25	5	6	6	7	9	11
4	9	10	10	10	10	10	6	7	7	7	8	8
7	13	14	14	14	14	14	7	8	8	8	8	8
10	16	17	17	17	17	17	8	9	9	9	9	9

TABLE 4.6
Example I. *NIPG-H: the number of iterations for multigrid with LU-SGS smoother as solver and preconditioner.*

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	4	4	4	4	4	4	3	4	4	4	4	5
5	5	6	6	6	6	6	4	5	5	5	5	5
10	7	8	8	8	8	8	6	6	6	6	6	6

Table 4.7

Example I. *NIPG-H: the number of iterations for multigrid with block-Jacobi smoother as solver and preconditioner.*

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	5	7	8	8	8	8	4	6	6	6	6	6
5	10	11	11	11	11	11	6	7	7	7	7	7
10	15	16	16	16	16	16	8	9	9	9	9	9

Finally, we test the performance of multigrid without the local correction operator in step 3 of Algorithm 3.1. The results are shown for HDG with $\tau = 1/h_{min}$ and the block-Jacobi smoother in Table 4.8. As can be seen, for $p \leq 3$, multigrid as a solver and also as a preconditioner fails to converge for fine mesh sizes. For $p \geq 4$ we see scalable results although the number of iterations is slightly greater than that in Table 4.3 with local correction. We also see an odd-even behavior, with even orders giving better iteration counts than odd ones. For $\tau = 1$, we observed similar results and hence these are not shown. For the hybridized IPDG schemes, with NIPG-H and IIPG-H from $p \geq 2$ onward we observed scalable results, whereas for SIPG-H for all orders we obtained scalability without local correction. The iteration counts are again slightly more than those obtained with local correction. With increase in order, since the block-Jacobi smoother gets stronger, even without the local correction operator it is sufficient to give scalable results. However, in order to provide more robustness and since the local correction operator is inexpensive, in the following test cases we always include it in step 3 of Algorithm 3.1.

Table 4.8

Example I. *HDG with stabilization $\tau = 1/h_{min}$: the number of iterations for multigrid with block-Jacobi smoother as solver and preconditioner without local correction.*

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	*	*	*	*	*	*	8	18	56	*	*	*
2	8	13	21	26	27	28	6	7	9	11	11	11
3	16	60	*	*	*	*	7	11	20	39	97	*
4	10	11	12	13	13	13	7	7	8	8	8	8
7	15	20	23	25	26	26	9	10	11	11	11	11
10	18	19	19	20	20	20	9	10	10	10	10	10

In summary, for HDG with $\tau = 1/h_{min}$ and multigrid preconditioned GMRES, almost perfect *hp*-scalability has been observed for both LU-SGS and block-Jacobi smoothers. It is also true for point-Jacobi as well as Chebyshev accelerated point-Jacobi smoothers as shown in [48]. For $\tau = 1$, the *hp*-scalability holds with only block smoothers together with high solution orders. All the hybridized IPDG schemes give scalable results with $\tau = (p+1)(p+2)/h_{min}$ and the number of iterations is very similar to that of HDG with $\tau = 1/h_{min}$. Even though LU-SGS seems to be a better smoother in terms of the number of iterations, block-Jacobi is more convenient from the implementation point of view (especially on parallel computing sytems), and for that reason we show numerical results only for the block-Jacobi smoother from now on. Since we observed the Chebyshev acceleration to block-Jacobi does not seem to improve the performance of multigrid by a significant margin, we will not use it in the subsequent sections. We would like to mention that the performance of our multigrid approach in terms of iteration counts is better than or at least comparable to other existing multigrids proposed in the literature [14, 18, 29, 31]. Since the time to solution depends on many other factors such as the choice of smoother, the number of smoothing steps, and the architecture of the machine, we are not able to comment on that aspect at this moment. However, in future work we plan to carry out a detailed study between different multigrid algorithms for hybridized methods with respect to simulation of challenging problems.

4.2. Example II: Unstructured mesh. The goal of this section is to test the robustness of the multigrid algorithm for a highly unstructured mesh. To that end consider the mesh in Figure 3.2, which consists of 6699 simplices with an order of magnitude variation in the diameter of elements, i.e., $h_{max} \approx 10h_{min}$. Unlike structured meshes, for unstructured meshes with local clustering of elements (local refinements) it is not straightforward to select the number of levels and the "best" coarsening strategy that well balances the number of elements (in the finer level) for each macro-element. Ideally, given a number of levels we wish to minimize the number of multigrid/GMRES iterations. Here, we compare two coarsening strategies with the same number of levels and study the performance of multigrid as a solver and as a preconditioner. In our future work, we will explore the coarsening strategies similar to the ones in [42, 43]. Figures 3.3 and 3.4 show different levels corresponding to the two coarsening strategies. The total number of levels in both the strategies is seven and the last level corresponds to the original fine mesh as shown in Figure 3.2. For solution orders $p > 1$, again, we first restrict the residual to $p = 1$ and then carry out the geometric coarsening. There are seven macro-elements in the first level for coarsening strategy 1 and four macro-elements for strategy 2.

We take the zero Dirichlet boundary condition, $\mathbf{K} = \mathbf{I}$ ($\mathbf{I}$ is the identity), and $f = 1$ for this example. We now study the performance of HDG and hybridized IPDG schemes. In Table 4.9, we compare the number of iterations taken for HDG with $\tau_1 = 1$ and $\tau_2 = 1/h_{min}$ and solution orders $p = 1, \dots, 8$. As can be seen, for $p \in \{1, 2, 3\}$, using τ_1 takes fewer iterations, and for $p \geq 4$ both stabilization parameters require almost the same iteration counts. For coarsening strategy 2, since the coarsening happens slightly more aggressively (i.e., fewer macro-elements at any level) the iteration counts in general are higher. However, using multigrid as a preconditioner for GMRES the difference in the iteration counts for the two strategies is negligible.

In Table 4.10, we consider the NIPG-H scheme with $\tau_1 = 1/h_{min}$ and $\tau_2 = (p+1)(p+2)/h_{min}$. As can be seen, the multigrid solver diverges for many values

TABLE 4.9
Example II. *HDG with $\tau_1 = 1$, $\tau_2 = 1/h_{min}$: the number of iterations for multigrid as solver and preconditioner for both coarsening strategies.*

	MG as solver				MG with GMRES			
p	Coarsening strategy 1		Coarsening strategy 2		Coarsening strategy 1		Coarsening strategy 2	
	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2
1	25	*	41	108	10	21	13	18
2	20	63	33	37	10	12	12	13
3	23	24	39	41	10	11	13	14
4	27	27	43	45	11	11	14	14
5	30	30	48	49	11	12	15	15
6	33	33	51	52	12	12	15	15
7	36	36	55	55	13	13	16	16
8	38	38	58	58	13	13	16	16

Table 4.10
Example II. *NIPG-H with $\tau_1 = 1/h_{min}$, $\tau_2 = (p+1)(p+2)/h_{min}$: the number of iterations for multigrid as solver and preconditioner for both coarsening strategies.*

	MG as solver				MG with GMRES			
p	Coarsening strategy 1		Coarsening strategy 2		Coarsening strategy 1		Coarsening strategy 2	
	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2
1	16	*	29	*	9	133	11	112
2	*	*	*	*	136	*	*	*
3	*	*	*	*	42	42	49	31
4	22	*	37	*	10	71	13	62
5	25	67	41	95	11	16	14	19
6	28	*	46	*	11	28	14	27
7	31	71	49	99	12	17	15	21
8	34	121	53	106	12	20	15	21

of solution order p. The iterations for τ_1 are less than that for τ_2. We observe that the iteration counts of multigrid preconditioned GMRES with τ_1, except for solution orders $p = 2$ and $p = 3$ (which require more iterations), are similar to those for HDG. When multigrid is used as a preconditioner, except for solution orders equal to two and three, both coarsening strategies give similar iteration counts with τ_1. It turns out that the iteration counts for IIPG-H and SIPG-H are higher than that of HDG and NIPG-H and hence are omitted for brevity.

Within the settings of this example we conclude that *multigrid*, both *as a solver and as a preconditioner, is more effective for HDG than for hybridized IPDG schemes.* It is also relatively more robust with respect to the coarsening strategies and values of stabilization parameter when used as a preconditioner.

4.3. Example III: Smoothly varying permeability. In this example we consider a smoothly varying permeability of the form

$$\mathbf{K} = \kappa \mathbf{I},$$

where $\kappa = 1 + 0.5 \sin(2\pi x) \cos(3\pi y)$. The domain considered is a circle and we take the exact solution again to be of the form $q = xye^{x^2 y^3}$. The forcing and the Dirichlet boundary condition are chosen corresponding to the exact solution. The number of levels and the corresponding number of triangular elements in the multigrid hierarchy are shown in Table 4.11.

Table 4.11
Example III: *The multigrid hierarchy.*

Levels	2	3	4	5	6	7
Elements	84	348	1500	6232	25344	102160

Table 4.12 shows the number of multigrid and GMRES iterations for HDG with $\tau = 1$. Note that perfect h-scalability with multigrid as a solver is not observed, and this is mainly due to the fact that the number of elements between successive levels is not exactly increased/decreased by a constant.[1] Again, we see a little growth in the number of iterations as p increases for the multigrid solver, while for GMRES with multigrid preconditioner we obtain almost perfect hp-scalability. We have also conducted numerical examples with $\tau = 1/h_{min}$, $\tau = \kappa/h_{min}$ and observed that the iteration counts (not shown here) are similar to those with $\tau = 1$.

[1]This, as argued before, is not trivial for unstructured meshes.

TABLE 4.12
Example III. *HDG with stabilization $\tau = 1$: the number of iterations for multigrid as solver and preconditioner.*

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	18	20	23	21	24	27	9	10	11	11	11	12
2	13	15	18	17	19	20	9	9	10	10	11	11
3	16	18	21	20	23	24	10	10	11	11	11	12
4	18	21	23	23	25	27	10	11	12	12	12	12
5	20	24	26	25	28	29	11	12	12	12	13	13
6	22	27	28	28	31	32	11	13	13	13	13	13
7	24	30	30	30	33	34	12	13	13	13	14	14
8	26	32	32	32	35	36	12	13	14	14	14	14

Next we consider the IIPG-H scheme with $\tau = \kappa(p+1)(p+2)/h_{min}$. Table 4.13 shows that the iteration counts are comparable with HDG results in Table 4.12 for GMRES with multigrid preconditioner. The multigrid solver for IIPG-H, on the other hand, has slightly fewer iteration counts. An extra penalization factor of 1.5 is necessary to obtain well-posed linear systems for SIPG-H, but otherwise the corresponding results for NIPG-H and SIPG-H are similar and hence are omitted.

TABLE 4.13
Example III. *IIPG-H with stabilization $\tau = \kappa(p+1)(p+2)/h_{min}$: the number of iterations for multigrid as solver and preconditioner.*

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	14	15	17	16	18	20	8	9	10	10	10	10
2	12	13	16	15	17	18	8	9	9	9	10	10
3	14	16	18	18	20	21	9	10	10	10	11	11
4	17	19	21	21	23	24	10	10	11	11	12	12
5	19	21	23	23	25	26	11	11	12	12	12	12
6	21	24	26	25	28	29	11	12	12	12	13	13
7	22	26	27	27	30	31	11	13	13	13	13	13
8	24	29	29	29	32	33	12	13	13	13	13	14

4.4. Example IV: Highly discontinuous permeability. In this example, we test the robustness of the multigrid solver on a highly discontinuous and spatially varying (six orders of magnitude) permeability field given by $\mathbf{K} = \kappa\mathbf{I}$ with

$$\kappa = \begin{cases} 10^6, & (x,y) \in (0,0.56)\times(0,0.56) \quad \text{or} \quad (x,y)\in(0.56,1)\times(0.56,1), \\ 1 & \text{otherwise,} \end{cases}$$

in a unit square.

We choose zero Dirichlet boundary condition and $f = 1$. In Table 4.14, we study the performance of multigrid as a solver and as a preconditioner for GMRES using HDG with $\tau = \kappa/h_{min}$. The number of elements and the number of levels are the same as in example I. Similar to the previous examples, almost perfect h- and p-scalabilities are observed with multigrid preconditioned GMRES, whereas the number of iterations increases with the solution order p for the multigrid solver. For $\tau = 1/h_{min}$ and $\tau = 1$ the multigrid solver takes too many iterations to converge for finer meshes, and for that reason we report the iteration counts only for multigrid preconditioned GMRES

Table 4.14
Example IV. *HDG with stabilization $\tau = \kappa/h_{min}$: the number of iterations for multigrid as solver and preconditioner.*

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	6	8	8	9	9	9	2	5	5	5	5	5
2	6	7	9	10	10	11	3	4	5	5	5	5
3	8	9	10	10	10	10	3	5	6	6	5	5
4	10	11	11	11	11	11	4	6	6	6	6	5
5	11	13	13	13	13	13	4	6	7	7	6	6
6	13	14	14	14	14	14	4	6	7	7	7	6
7	14	15	16	16	15	15	5	7	7	7	7	7
8	15	16	17	17	17	17	5	7	8	8	7	7

Table 4.15
Example IV. *HDG with stabilizations $\tau = 1/h_{min}$ and $\tau = 1$: the number of iterations for multigrid preconditioned GMRES.*

p	$\tau = 1/h_{min}$						$\tau = 1$					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	3	5	6	8	11	5	2	5	8	11	15	15
2	3	5	6	5	5	5	3	5	6	7	8	11
3	3	5	6	6	5	5	3	4	5	6	7	8
4	4	5	6	6	6	5	4	5	6	6	7	7
5	4	5	7	7	6	6	4	5	7	7	6	6
6	4	6	7	7	7	6	4	6	7	7	7	6
7	4	6	7	7	7	7	4	6	7	7	7	7
8	4	6	8	8	7	7	4	6	8	8	7	7

in Table 4.15. Clearly, the number of GMRES iterations is almost insensitive to the values of the stabilization parameter τ, and the results for $\tau = 1/h_{min}$ in columns 2–7 of Table 4.15 are very similar to columns 8–13 of Table 4.14 for $\tau = \kappa/h_{min}$. Columns 8–13 of Table 4.15 show that, again with $\tau = 1$, high-order solutions provide both h- and p-scalabilities and the results in these cases are similar to those with $\tau = 1/h_{min}$ (see the last four rows of Table 4.15).

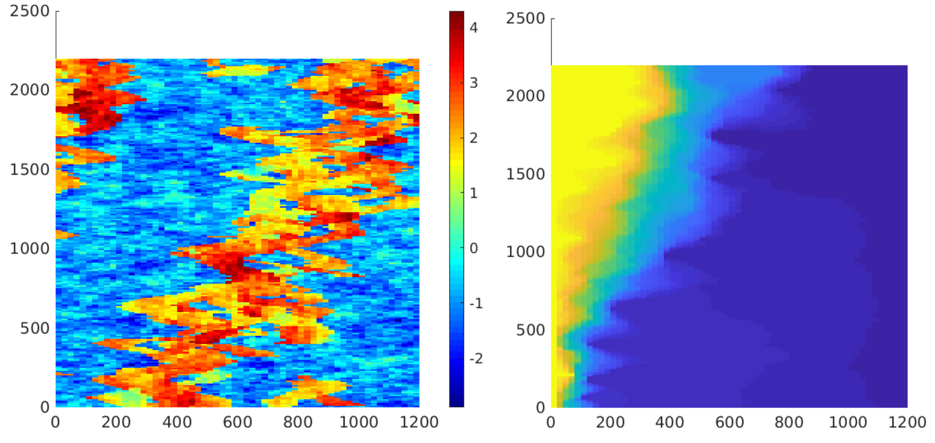
Table 4.16 shows the performance of our multigrid algorithm as a solver and as a preconditioner for the SIPG-H scheme with $\tau = \kappa(p+1)(p+2)/h_{min}$. The results

Table 4.16
Example IV. *SIPG-H with $\tau = \kappa(p+1)(p+2)/h_{min}$: the number of iterations for multigrid as solver and preconditioner.*

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	8	9	10	11	11	12	3	6	6	6	6	6
2	6	8	9	9	10	10	4	5	5	5	5	5
3	9	10	10	10	10	11	4	6	6	6	6	5
4	10	11	12	12	12	12	4	6	7	6	6	6
5	12	13	13	13	13	13	4	6	7	7	7	6
6	13	14	15	15	15	15	5	7	7	7	7	7
7	14	16	16	16	16	16	5	7	8	8	7	7
8	15	17	17	17	17	17	5	7	8	8	7	7

for NIPG-H and IIPG-H are almost identical and are omitted. As can be observed, as a preconditioner for GMRES the multigrid algorithm is both h- and p-scalable for HDG as well as the hybridized IPDG methods.

4.5. Example V: SPE10 test case. In this example, we consider the benchmark problem Model 2 from the Tenth Society of Petroleum Comparative Solution Project (SPE10) [16]. We consider the permeability field $\mathbf{K} = \kappa\mathbf{I}$, where κ corresponding to the 75th layer is shown on the left of Figure 4.1. The permeability field varies by six orders of magnitude and is highly heterogeneous, which gives rise to extremely complex velocity fields. The domain is 1200×2200 $[ft^2]$. The mesh has 60×220 quadrilateral elements and the element edges align with the discontinuities in the permeability. We choose $f = 0$, which corresponds to no source or sink. For the boundary conditions we take the pressure (q) on the left and right faces to be 1 and 0, respectively. On the top and bottom faces no flux boundary condition $\mathbf{u}\cdot\mathbf{n} = 0$ is applied. From extensive numerical examples we have observed that the multigrid hierarchy with seven levels results in the fewest GMRES iterations. From numerical examples I–IV, we see that the HDG method is relatively the most robust and scalable. In addition, it provides simultaneous approximations for both velocity and pressure. Thus, we consider only HDG for this example.



(a) Permeability field (b) Pressure field

FIG. 4.1. *Example* V: *Permeability (κ) in log scale (left) and pressure field (right) using solution order $p = 1$.*

Since multigrid as a solver either converges very slowly or diverges for a number of cases in this example, we report results exclusively for the multigrid preconditioned GMRES. The pressure field is shown on the right of Figure 4.1 for $p = 1$. In Table 4.17 are the number of iterations for solution orders $p \in \{1, 2, 3, 4\}$ with $\tau = 1/h_{min}$ and $\tau = 1$. The second row shows that, for $p = 2$, the iterations are much larger compared to other solution orders. At the time of writing, we had not yet found the reason for this behavior. For other solution orders, using $\tau = 1$ results in more iterations for $p = 1$ and fewer for $p = 3, 4$ compared to $\tau = 1/h_{min}$. Again, we emphasize that in this example we completely avoid upscaling of the permeability field as our multigrid algorithm is based only on the fine scale DtN maps.

The results in columns 2 and 3 of Table 4.17 correspond to geometrically increasing smoothing steps with respect to levels as explained in section 3.3. In columns 4

Table 4.17
Example V. *HDG with stabilizations $\tau = 1/h_{min}$, $\tau = 1$: the number of iterations for multigrid preconditioned GMRES with variable smoothing and constant smoothing.*

p	Varying smoothing		2 pre- and postsmoothing	
	$\tau = 1/h_{min}$	$\tau = 1$	$\tau = 1/h_{min}$	$\tau = 1$
1	47	78	51	80
2	81	*	72	*
3	58	40	56	48
4	48	39	49	45

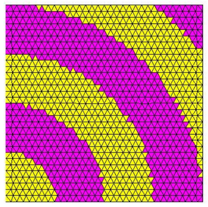
and 5, we simply take two pre- and postsmoothing steps in all levels and, as can be seen, the iteration counts are marginally different compared to those resulting from the increasing smoothing steps. This implies that we can achieve similar accuracy with less computational cost using constant (here two) pre- and postsmoothing steps in all levels.

4.6. Example VI: Multinumerics. The goal of this section is to demonstrate that the proposed multigrid algorithm can handle different fine scale DtN maps (multinumerics) on unstructured grids without explicit upscaling for macro-elements. In the following, we choose a combination of RT1 and NIPG as a representative for other combinations of hybridized methods. We expect similar results using other combinations, as our multigrid algorithm operates on the trace system. Also we would like to point out that the other multigrid algorithms for hybridized methods [14, 18, 29, 31] may also work for the case of multinumerics. However, to the best of our knowledge we are not able to find such a study yet and it is beyond the current scope of this paper. In our future work we intend to compare these different algorithms along with our approach in the context of multinumerics.

We consider the permeability field

$$\mathbf{K} = (1 + 1/2 \sin(4\pi x) \cos(3\pi y))\mathbf{I},$$

and $f = 1$. We consider the unit square and use either RT1 mixed finite elements or first-order NIPG discontinuous Galerkin as shown in Figure 4.2 (left). We plot the finest mesh in the left of Figure 4.2 and a coarsening strategy with four levels in Figure 4.3. We point out that (many or all) macro-elements on the coarser grids contain both RT1 and NIPG fine grid elements.



Levels	3	4	5	6	7
Elements	120	496	2016	8128	32640
Iterations	9	10	9	9	9

Fig. 4.2. *Example* VI: *On the left, NIPG-H elements are purple and RT*1 *elements are yellow. On the right are the number of iterations for the multigrid solver using multinumerics.*

On the right of Figure 4.2 are the iteration counts using the multigrid solver. It can be observed that the performance, i.e., the number of iterations, using multinumerics is nearly identical to that for NIPG-H in section 4.1 despite the fact that different numerical methods are used throughout the domain and the macro-elements

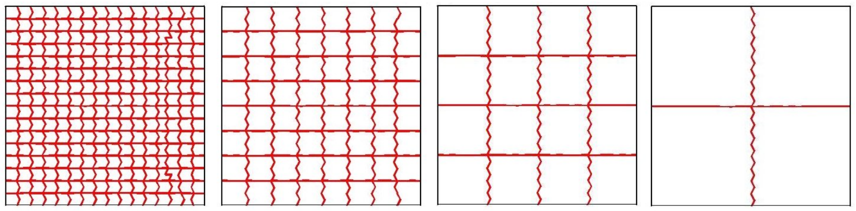


FIG. 4.3. *Example* VI: *Coarsening strategy.*

are neither triangles nor quadrilaterals. This flexibility is one of the highlighted characteristics of our multigrid algorithm.

5. Conclusion.

We have proposed a unified DtN-based geometric multigrid algorithm for hybridized high-order finite element methods. Our approach differs significantly from the previous attempts in the sense that the intergrid transfer operators are physics-based energy-preserving and the coarse grid operators are discretized DtN maps on every level. These operators completely avoid upscaling of parameters, such as permeability and source, to coarse scales. As our numerical results have indicated, they also allow for multinumerics and variable coefficients without deteriorating the performance of the multigrid algorithm. We have presented several numerical examples with HDG methods and hybridized IPDG methods and reported several observations. For HDG methods, with stabilization $\tau = 1$ or $1/h_{min}$ (depending on the example), we obtain almost perfect scalability in mesh size and solution order using multigrid preconditioned GMRES. For all the other hybridized IPDG methods, with a stabilization of the form $\tau = \mathcal{O}\left(p^2/h_{min}\right)$ similar scalability is observed. However, the multigrid algorithm applied to IPDG seems to be less robust compared to HDG with respect to stabilization parameters and coarsening strategies, especially for highly unstructured meshes. We have also reported scalable results for the proposed multgrid method when applying to multinumeric discretizations with mixed methods in certain parts of the domain and DG methods in other parts. We are currently extending our multigrid algorithm to convection-diffusion, Stokes, Oseen, and incompressible resistive magnetohydrodynamic equations. Research is ongoing to provide a rigorous analysis of the proposed multigrid algorithm and we will report our findings elsewhere in the near future.

Acknowledgment. The authors would like to thank the anonymous referees for their critical and useful comments, which significantly improved the paper.

REFERENCES

[1] D. N. Arnold and F. Brezzi, *Mixed and nonconforming finite element methods: Implementation, postprocessing and error estimates*, ESAIM Math. Model. Numer. Anal., 19 (1985), pp. 7–32.

[2] D. Braess, W. Dahmen, and C. Wieners, *A multigrid algorithm for the mortar finite element method*, SIAM J. Numer. Anal., 37 (1999), pp. 48–69.

[3] D. Braess and R. Verfurth, *Multigrid methods for nonconforming finite element methods*, SIAM J. Numer. Anal., 27 (1990), pp. pp. 979–986.

[4] J. Bramble, R. Ewing, J. Pasciak, and J. Shen, *The analysis of multigrid algorithms for cell centered finite difference methods*, Adv. Comput. Math., 5 (1996), pp. 15–29.

[5] J. H. Bramble, *Multigrid Methods*, Chapman & Hall/CRC Res. Notes Math. 294, CRC Press, Boca Raton, FL, 1993.

[6] J. H. Bramble, J. E. Pasciak, and J. Xu, *The analysis of multigrid algorithms with nonnested spaces or noninherited quadratic forms*, Math. Comp., 56 (1991), pp. 1–34.

[7] J. H. Bramble and J. E. Pasicak, *Uniform convergence estimates for multigrid v-cycle algorithms with less than full elliptic regularity*, Contemp. Math., 157 (1994), pp. 17–26.

[8] S. C. Brenner, *A multigrid algorithm for the lowest-order Raviart-Thomas mixed triangular finite element method*, SIAM J. Numer. Anal., 29 (1992), pp. 647–678.

[9] S. C. Brenner, *Convergence of nonconforming multigrid methods without full elliptic regularity*, Math. Comp., 68 (1999), pp. 25–53.

[10] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer Ser. Comput. Math. 15, Springer, New York, 2012.

[11] T. Bui-Thanh, *From Godunov to a unified hybridized discontinuous Galerkin framework for partial differential equations*, J. Comput. Phys., 295 (2015), pp. 114–146.

[12] T. Bui-Thanh, *From Rankine-Hugoniot condition to a constructive derivation of HDG methods*, in Spectral and High Order Methods for Partial Differential Equations, Lect. Notes Comput. Sci. Eng. 106, Springer, New York, 2015, pp. 483–491.

[13] T. Bui-Thanh, *Construction and analysis of HDG methods for linearized shallow water equations*, SIAM J. Sci. Comput., 38 (2016), pp. A3696–A3719.

[14] L. Chen, J. Wang, Y. Wang, and X. Ye, *An auxiliary space multigrid preconditioner for the weak Galerkin method*, Comput. Math. Appl., 70 (2015), pp. 330–344.

[15] Z. Chen, *Equivalence between and multigrid algorithms for nonconforming and mixed methods for second-order elliptic problems*, East-West J. Numer. Math., 4 (1996), pp. 1–33.

[16] M. A. Christie and M. J. Blunt, *Tenth SPE comparative solution project: A comparison of upscaling techniques*, in Proceedings of the SPE Reservoir Simulation Symposium, Society of Petroleum Engineers, 2001.

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

[18] B. Cockburn, O. Dubois, J. Gopalakrishnan, and S. Tan, *Multigrid for an HDG method*, IMA J. Numer. Anal., 34 (2014), pp. 1386–1425.

[19] B. Cockburn and J. Gopalakrishnan, *A characterization of hybridized mixed methods for second order elliptic problems*, SIAM J. Numer. Anal., 42 (2004), pp. 283–301.

[20] 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 (2009), pp. 1319–1365.

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

[22] Z. De and M. Paulus, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1–27.

[23] J. E. Dendy, *Black box multigrid*, J. Comput. Phys., 48 (1982), pp. 366–386.

[24] M. S. Fabien, M. G. Knepley, and B. M. Rivière, *A hybridizable discontinuous Galerkin method for two-phase flow in heterogeneous porous media*, Internat. J. Numer. Methods Engrg., 116 (2018), pp. 161–177.

[25] B. Fraeijs de Veubeke, *Displacement and equilibrium models in the finite element method*, in Stress Analysis, Witney, New York, 1965, pp. 145–195.

[26] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.

[27] V. Ginting, G. Lin, and J. Liu, *On application of the weak Galerkin finite element method to a two-phase model for subsurface flow*, J. Sci. Comput., 66 (2016), pp. 225–239.

[28] V. Girault, S. Sun, M. F. Wheeler, and I. Yotov, *Coupling discontinuous Galerkin and mixed finite element discretizations using mortar finite elements*, SIAM J. Numer. Anal., 46 (2008), pp. 949–979.

[29] J. Gopalakrishnan and S. Tan, *A convergent multigrid cycle for the hybridized mixed method*, Numer. Linear Algebra Appl., 16 (2009), pp. 689–714.

[30] B. T. Helenbrook and H. L. Atkins, *Application of p-multigrid to discontinuous Galerkin formulations of the poisson equation*, AIAA J., 44 (2006), pp. 566–575.

[31] M. Kronbichler and W. A. Wall, *A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers*, SIAM J. Sci. Comput., 40 (2018), pp. A3423–A3448.

[32] D. Y. Kwak, *V-cycle multigrid for cell-centered finite differences*, SIAM J. Sci. Comput., 21 (1999), pp. 552–564.

[33] L. Mu, J. Wang, and X. Ye, *A hybridized formulation for the weak Galerkin mixed finite element method*, J. Comput. Appl. Math., 307 (2016), pp. 335–345.

[34] C. Oltean and M. A. Buès, *Coupled groundwater flow and transport in porous media. a conservative or non-conservative form?*, Transp. Porous Media, 44 (2001), pp. 219–246.

[35] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations: Numerical Mathematics and Scientific Computation*, Oxford University Press, New York, 1999.

[36] A. Reusken, *Multigrid with matrix-dependent transfer operators for a singular perturbation problem*, Computing, 50 (1993), pp. 199–211.

[37] A. Reusken, *Multigrid with matrix-dependent transfer operators for convection-diffusion problems*, in Multigrid Methods IV, Springer, New York, 1994, pp. 269–280.

[38] B. Rivière, M. F. Wheeler, and V. Girault, *A priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems*, SIAM J. Numer. Anal., 39 (2001), pp. 902–931.

[39] J. E. Roberts and J.-M. Thomas, *Mixed and hybrid methods*, in Finite Element Methods, Handb. Numer. Anal. Z, Elsevier, Amsterdam, 1991, pp. 523–639.

[40] A. Samii, C. Michoski, and C. Dawson, *A parallel and adaptive hybridized discontinuous galerkin method for anisotropic nonhomogeneous diffusion*, Comput. Methods Appl. Mech. Engrg, 304 (2016), pp. 118–139.

[41] U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid*, Academic Press, New York, 2000.

[42] V. Venkatakrishnan and D. J. Mavriplis, *Agglomeration multigrid for the Euler equations*, in Fourteenth International Conference on Numerical Methods in Fluid Dynamics, Springer, New York, 1995, pp. 178–182.

[43] V. Venkatakrishnan, D. J. Mavriplis, and M. J. Berger, *Unstructured multigrid through agglomeration*, in Sixth Annual Copper Mountain Conference on Multigrid Methods, 1993.

[44] C. Wagner, W. Kinzelbach, and G. Wittum, *Schur-complement multigrid*, Numer. Math., 75 (1997), pp. 523–545.

[45] C. Waluga, *Analysis of Hybrid Discontinuous Galerkin Methods for Incompressible Flow Problems*, Diplomingenieur thesis, RWTH Aachen University, Germany, 2012.

[46] M. F. Wheeler and M. Peszyńska, *Computational engineering and science methodologies for modeling and simulation of subsurface applications*, Adv. Water Res., 25 (2002), pp. 1147–1173.

[47] M. F. Wheeler and I. Yotov, *Multigrid on the interface for mortar mixed finite element methods for elliptic problems*, Comput. Methods Appl. Mech. Engrg, 184 (2000), pp. 287–302.

[48] T. Wildey, S. Muralikrishnan, and T. Bui-Thanh, *Unified Geometric Multigrid Algorithm for Hybridized High-Order Finite Element Methods*, preprint, arXiv:1811.09909, 2018.

[49] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.