SIAM J. SCI. COMPUT.
Vol. 41, No. 6, pp. A3913–A3937

EFFICIENT OPERATOR-COARSENING MULTIGRID SCHEMES FOR LOCAL DISCONTINUOUS GALERKIN METHODS*

DANIEL FORTUNATO†, CHRIS H. RYCROFT†‡, AND ROBERT SAYE‡

Abstract. An efficient *hp*-multigrid scheme is presented for local discontinuous Galerkin (LDG) discretizations of elliptic problems, formulated around the idea of separately coarsening the underlying discrete gradient and divergence operators. We show that traditional multigrid coarsening of the primal formulation leads to poor and suboptimal multigrid performance, whereas coarsening of the flux formulation leads to essentially optimal convergence and is equivalent to a purely geometric multigrid method. The resulting operator-coarsening schemes do not require the entire mesh hierarchy to be explicitly built, thereby obviating the need to compute quadrature rules, lifting operators, and other mesh-related quantities on coarse meshes. We show that good multigrid convergence rates are achieved in a variety of numerical tests on two-dimensional and three-dimensional uniform and adaptive Cartesian grids, as well as for curved domains using implicitly defined meshes and for multiphase elliptic interface problems with complex geometry. Extensions, e.g., to non-LDG discretizations and fully unstructured meshes, are briefly discussed.

Key words. discontinuous Galerkin methods, multigrid methods, elliptic interface problems, implicitly defined meshes

AMS subject classifications. 65N55, 65N30, 65F08

DOI. 10.1137/18M1206357

1. Introduction.

Discontinuous Galerkin (DG) methods have gained broad popularity in recent years. They are well-suited to *hp*-adaptivity, provide high-order accuracy, and can be applied to a wide range of problems on complex geometries with unstructured meshes. Although DG methods were first applied to the discretization of hyperbolic conservation laws, they have been extended to handle elliptic problems and diffusive operators in a unified framework [7]. Such methods include the symmetric interior penalty (SIP) method [21, 6], the Bassi–Rebay (BR1, BR2) methods [9, 10], the local discontinuous Galerkin (LDG) method [20], the compact discontinuous Galerkin (CDG) method [31], the line-based discontinuous Galerkin method [32], and the hybridizable discontinuous Galerkin (HDG) method [18]. In particular, the development of efficient solvers for DG discretizations of elliptic problems is an active area of research.

Among the panoply of DG methods for elliptic problems, the LDG method is a popular choice: it is accurate, stable, simple to implement, and extendable to higher-order derivatives [44]. Additionally, on Cartesian grids it has been shown to be superconvergent [19]. The LDG method results in symmetric positive (semi)definite

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 9, 2018; accepted for publication (in revised form) August 30, 2019; published electronically December 17, 2019.

https://doi.org/10.1137/18M1206357

Funding: The first author was supported by the National Defense Science and Engineering Graduate Fellowship. The second and third authors were supported by the Applied Mathematics Program of the U.S. DOE Office of Advanced Scientific Computing Research under contract DE-AC02-05CH11231. Some computations used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231.

†John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 (dfortunato@g.harvard.edu, chr@seas.harvard.edu).

‡Mathematics Group, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (rsaye@lbl.gov).

discretizations which are well-suited to solution by efficient iterative methods. In particular, the multigrid method has emerged as a natural candidate due to its success in the continuous finite element and finite difference communities, both as a standalone solver and as a preconditioner for the conjugate gradient (PCG) method. However, direct application of standard multigrid techniques to DG discretizations of elliptic problems can result in suboptimal performance when inherited bilinear forms are employed [2, 24], and much work has gone into developing specialized smoothers and coarse-correction methods to remedy this issue, even on Cartesian grids [27, 22].

When a mesh hierarchy is available, geometric h-multigrid is a natural choice of solver. Error estimates have been derived for a multilevel interior penalty (IP) method on unstructured meshes, yielding convergence factors in the range $\rho \approx 0.3$–0.5 for Poisson's equation [24, 12]; the method has also been applied to adaptively refined Cartesian grids with similar results [28]. Subsequent work describes how the multilevel IP method [24] can be used as a preconditioner for the LDG method—both for the Schur complement system (using conjugate gradient) and for the saddle-point system (using GMRES)—resulting in a bounded condition number with respect to mesh size h [27]. More recent work for LDG and IP uses a multigrid W-cycle on nested [2] and agglomerated [3] unstructured meshes; however, these results indicate poor convergence factors of $\rho \approx 0.8$–0.9 even with many smoothing steps. On nonnested polygonal meshes, h-independent iteration counts with convergence factors $\rho \approx 0.2$–0.3 are achieved for SIP using an additive Schwarz smoother with 3–8 smoothing steps per V-cycle [5].

A popular choice for high-order DG methods is p- or hp-multigrid [25, 23, 29, 8], where p refers to coarsening the polynomial degree p in the multilevel hierarchy, and hp refers to some combined strategy of coarsening the mesh size h as well as the polynomial degree p of the underlying discretization. Using factor-of-two coarsening in p with an element Jacobi smoother, convergence factors of $\rho \approx 0.5$ were achieved for Laplace's equation with $p \leq 4$ [25]. Fidkowski et al. [23] used a line smoother with sequential coarsening in p that gave similar results for convection–diffusion problems, though the performance degraded as $h \to 0$. A method employing an overlapping Schwarz smoother with factor-of-two coarsening was used with PCG on LDG discretizations up to $p = 32$ [39]; this method exhibited good convergence factors of $\rho \leq 0.1$ on high-aspect-ratio Cartesian grids, at the cost of an expensive smoother.

Algebraic multigrid methods have also been applied to DG discretizations of elliptic problems. A hierarchy of operators can be defined by agglomerating neighboring unknowns based on smoothed aggregation (SA), resulting in average convergence factors of $\rho \approx 0.4$ and $\rho \approx 0.2$ for the bilinear BR2 and SIP methods, respectively [34]. An SA method employing energy minimization was used with PCG to achieve h-independent convergence factors of $\rho \approx 0.2$ for LDG discretizations, but performance degraded with increasing p [30]. A method based on unsmoothed aggregation was developed for the IP method using a coarse space consisting of continuous linear basis functions [11]; this method proved robust for multiphase problems with large jumps in ellipticity coefficient, but efficiency weakly degraded with mesh size. A related approach based on smoothed aggregation and low-order coarse grid correction yielded similar results [38].

Independent of the type of multigrid method, a particular fact to note—and something we believe underpins the difficulties in applying multigrid to DG—is that coarsening a fine-grid operator is not always the same as constructing that operator directly from the coarse grid. Indeed, it was noted by Antonietti, Sarti, and Verani [2] that for all stable and strongly consistent DG methods, "convergence cannot be

independent of the number of levels if inherited bilinear forms are considered (i.e., the coarse solvers are the restriction of the stiffness matrix constructed on the finest grid)." Furthermore they noted that noninherited forms must be employed for the multigrid method to be scalable. In the context of two-level methods, the reason for this loss of scalability is known [1]. In this paper, for the LDG method, we present a simple modification to traditional multigrid operator coarsening that yields optimal multigrid convergence and can be extended to other DG discretizations of elliptic problems. We confirm that traditional coarsening of the fine-mesh elliptic operator results in poor performance, and show that the coarsening of the saddle-point flux formation restores optimal multigrid efficiency. Our approach is equivalent to pure geometric multigrid but avoids the need to explicitly build the coarse mesh and its associated components, such as quadrature rules, Jacobian mappings, lifting operators, and face-to-element enumerations—as discussed, this holds benefit for a variety of intricate DG implementations where building the coarse mesh can be problematic. Nevertheless, we point out that in the pure geometric multigrid setting, quadrature-free DG methods [4] have recently been proposed which avoid the construction of coarse mesh quadrature rules.

The paper is structured as follows. In section 2, we formulate a general DG discretization of Poisson's equation and derive the LDG method through the appropriate choice of numerical flux. In section 3, we describe the construction of geometric *hp*-multigrid methods in the corresponding DG setting. In particular, we show that traditional operator coarsening can fail to create the coarse operator resulting from faithful rediscretization in a pure geometric multigrid setting for LDG methods, and present a modified coarsening strategy that remedies this. In section 4, we present numerical results for the standard and modified multigrid methods on uniform and adaptively-refined Cartesian grids in two dimensions (2D) and three dimensions (3D). We conclude with some examples of multiphase elliptic interface problems on implicitly defined meshes, which demonstrate good multigrid performance even on meshes with long and thin filaments as well as tiny and dispersed phase components.

2. DG formulation.

2.1. Model problem.

The model elliptic problem considered in this work is the Poisson problem

$$\begin{aligned} -\nabla^2 u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_D, \\ \nabla u \cdot \boldsymbol{n} &= h \quad \text{on } \Gamma_N, \end{aligned} \tag{2.1}$$

where Ω is a domain in $\mathbb{R}^d$, Γ_D and Γ_N denote the components of $\partial\Omega$ on which Dirichlet and Neumann boundary conditions are imposed, $\boldsymbol{n}$ is the outward unit normal to the boundary, and f, g, and h are given functions defined on Ω and its boundary.

2.2. DG for elliptic problems.

In order to apply a DG method to (2.1), we rewrite it as a first-order system by introducing the auxiliary variable $\boldsymbol{q} = \nabla u$ and writing the Laplacian as the divergence of $\boldsymbol{q}$ [7]:

$$\begin{aligned} \boldsymbol{q} &= \nabla u \quad \text{in } \Omega, \\ -\nabla \cdot \boldsymbol{q} &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_D, \\ \boldsymbol{q} \cdot \boldsymbol{n} &= h \quad \text{on } \Gamma_N. \end{aligned} \tag{2.2}$$

In this work, we mainly consider discretizations of (2.2) wherein the corresponding meshes arise from Cartesian grids, quad/octrees, or implicitly defined meshes of more complex curved domains (see sections 4.1, 4.3, and 4.4, respectively). As such, it is natural to adopt a tensor-product piecewise polynomial space. Let $\mathcal{E} = \bigcup_i E_i$ denote the set of elements of a mesh of Ω, let $p \geq 1$ be an integer, and define $\mathcal{Q}_p(E)$ to be the space of tensor-product polynomials of degree p on the element E. For example, $\mathcal{Q}_3$ is the space of bicubic (in 2D) or tricubic (in 3D) polynomials having 16 or 64 degrees of freedom, respectively. We define the corresponding spaces of discontinuous piecewise polynomials and vector fields on the mesh as

$$V_h(\mathcal{E}) = \left\{ v : \Omega \to \mathbb{R} \;\middle|\; v|_E \in \mathcal{Q}_p(E) \text{ for every } E \in \mathcal{E} \right\}, \tag{2.3}$$

$$V_h^d(\mathcal{E}) = \left\{ \boldsymbol{\omega} : \Omega \to \mathbb{R}^d \;\middle|\; \boldsymbol{\omega}|_E \in [\mathcal{Q}_p(E)]^d \text{ for every } E \in \mathcal{E} \right\}, \tag{2.4}$$

respectively. We denote by $(\cdot, \cdot)$ the natural L^2 inner product on V_h and by $\|\cdot\|$ the corresponding norm, $\|u\|^2 = (u, u)$, with analogous definitions for V_h^d.

In a DG method, both $\boldsymbol{q}$ and its divergence are defined weakly via numerical fluxes defined on each mesh face. The weak form of (2.2) consists of finding $(\boldsymbol{q}_h, u_h) \in V_h^d \times V_h$ such that

$$\int_E \boldsymbol{q}_h \cdot \boldsymbol{\omega} = -\int_E u_h \nabla \cdot \boldsymbol{\omega} + \int_{\partial E} \widehat{u}_h \, \boldsymbol{\omega} \cdot \boldsymbol{n}, \tag{2.5}$$

$$\int_E \boldsymbol{q}_h \cdot \nabla v - \int_{\partial E} \widehat{\boldsymbol{q}}_h \, v \cdot \boldsymbol{n} = \int_E f \, v \tag{2.6}$$

for all test functions $(\boldsymbol{\omega}, v) \in [\mathcal{Q}_p(E)]^d \times \mathcal{Q}_p(E)$ and for all $E \in \mathcal{E}$. The numerical fluxes $\widehat{\boldsymbol{q}}_h$ and $\widehat{u}_h$ are approximations to $\boldsymbol{q}_h$ and u_h, respectively, on each mesh face and define how the degrees of freedom in each element are coupled together.

To more succinctly describe the coupling between elements, the following standard notation is adopted. Consider two adjacent elements E^+ and E^- which share a face in $\mathcal{E}$. Let $\boldsymbol{n}^\pm$ denote the outward unit normals of $\partial E^\pm$ along the shared face and $(\boldsymbol{\omega}^\pm, v^\pm)$ denote the traces of $(\boldsymbol{\omega}, v) \in V_h^d \times V_h$ from $E^\pm$ on the shared face. The average $\{\!\{\cdot\}\!\}$ and jump $[\![\cdot]\!]$ operators on the shared face are then defined as

$$\begin{aligned} \{\!\{\boldsymbol{\omega}\}\!\} &= \tfrac{1}{2}(\boldsymbol{\omega}^+ + \boldsymbol{\omega}^-), & \{\!\{v\}\!\} &= \tfrac{1}{2}(v^+ + v^-), \\ [\![\boldsymbol{\omega}]\!] &= \boldsymbol{\omega}^+ \cdot \boldsymbol{n}^+ + \boldsymbol{\omega}^- \cdot \boldsymbol{n}^-, & [\![v]\!] &= v^+ \boldsymbol{n}^+ + v^- \boldsymbol{n}^-. \end{aligned}$$

On boundary faces, $(\boldsymbol{\omega}^-, v^-)$ shall refer to the traces of $(\boldsymbol{\omega}, v)$ from the corresponding element touching $\partial\Omega$.

2.3. The LDG method. The choice of numerical flux in (2.5)–(2.6) defines a DG method. Here we focus on the LDG method [20], which chooses numerical fluxes $\widehat{\boldsymbol{q}}_h$ and $\widehat{u}_h$ according to the general forms

$$\widehat{\boldsymbol{q}}_h = \begin{cases} \{\!\{\boldsymbol{q}_h\}\!\} + \boldsymbol{\beta}[\![\boldsymbol{q}_h]\!] - \tau_0 [\![u_h]\!] & \text{on any interior face,} \\ \boldsymbol{q}_h^- - \tau_D (u_h^- - g)\boldsymbol{n} & \text{on any face of } \Gamma_D, \\ h\,\boldsymbol{n} & \text{on any face of } \Gamma_N \end{cases} \tag{2.7}$$

and

$$\widehat{u}_h = \begin{cases} \{\!\{u_h\}\!\} - \boldsymbol{\beta} \cdot [\![u_h]\!] & \text{on any interior face,} \\ g & \text{on any face of } \Gamma_D, \\ u_h^- & \text{on any face of } \Gamma_N, \end{cases} \tag{2.8}$$

where $\boldsymbol{\beta}$ is a (possibly face-dependent) user-defined vector; for example, in a one-sided flux scheme, $\boldsymbol{\beta} = \pm\frac{1}{2}\boldsymbol{n}$. Here, the numerical flux $\widehat{\boldsymbol{q}}_h$ includes penalty stabilization terms; $\tau_0 \geq 0$ is a penalty parameter associated with interior faces and $\tau_D > 0$ is associated with Dirichlet boundary faces (if any). Generally, τ_0 must be strictly positive to ensure well-posedness of the discrete problem, but in some cases (e.g., on Cartesian grids with particular choices of $\boldsymbol{\beta}$), τ_0 can be set equal to zero [17]. If Γ_D is nonempty, τ_D must be positive to ensure well-posedness of the final discrete problem. To be consistent with the scaling of penalty parameters in other DG methods, we choose the penalty parameters to scale inversely with the element size h [16] so that $\tau_0 = \widetilde{\tau}_0/h$ and $\tau_D = \widetilde{\tau}_D/h$, where $\widetilde{\tau}_0$ and $\widetilde{\tau}_D$ are constants.[1] Furthermore, although arbitrarily small choices of $\widetilde{\tau}_0$ and $\widetilde{\tau}_D$ suffice to ensure well-posedness, later we show that a carefully considered choice of these values can greatly benefit multigrid performance (see section 4.2).

We first particularize (2.5) for the LDG method, which is the weak statement that $\boldsymbol{q} = \nabla u$. We slightly modify the weak form (2.5) by defining $\boldsymbol{q}_h \in V_h^d$ in strong-weak form,[2] such that

$$\int_E \boldsymbol{q}_h \cdot \boldsymbol{\omega} = \int_E \nabla u_h \cdot \boldsymbol{\omega} + \int_{\partial E} (\widehat{u}_h - u_h)\, \boldsymbol{\omega} \cdot \boldsymbol{n} \tag{2.9}$$

holds for every element $E \in \mathcal{E}$ and every test function $\boldsymbol{\omega} \in [\mathcal{Q}_p(E)]^d$. Upon summing (2.9) over every element of the mesh and using the definition of the numerical flux $\widehat{u}_h$ in (2.8), we have that, for any $\boldsymbol{\omega} \in V_h^d$,

$$\int_\Omega \boldsymbol{q}_h \cdot \boldsymbol{\omega} = \sum_{E\in\mathcal{E}} \int_E \nabla u_h \cdot \boldsymbol{\omega} - \int_{\Gamma_0} [\![u_h]\!] \cdot (\{\!\{\boldsymbol{\omega}\}\!\} + \boldsymbol{\beta}[\![\boldsymbol{\omega}]\!]) + \int_{\Gamma_D} (g - u_h^-)\boldsymbol{\omega}^- \cdot \boldsymbol{n}, \tag{2.10}$$

where Γ_0 denotes the union of all interior faces of $\mathcal{E}$. Define the following operators:

- Let $\nabla_h : V_h \to V_h^d$ be the *broken gradient operator* and $L : V_h \to V_h^d$ be the *lifting operator*, such that

$$\int_\Omega (\nabla_h u) \cdot \boldsymbol{\omega} = \sum_{E\in\mathcal{E}} \int_E \nabla u \cdot \boldsymbol{\omega},$$
$$\int_\Omega (Lu) \cdot \boldsymbol{\omega} = -\int_{\Gamma_0} [\![u]\!] \cdot (\{\!\{\boldsymbol{\omega}\}\!\} + \boldsymbol{\beta}[\![\boldsymbol{\omega}]\!]) - \int_{\Gamma_D} u^- \boldsymbol{\omega}^- \cdot \boldsymbol{n}$$

  holds for every $\boldsymbol{\omega} \in V_h^d$ and each $u \in V_h$.

- Define $J_D(g) \in V_h^d$ such that

$$\int_\Omega J_D(g) \cdot \boldsymbol{\omega} = \int_{\Gamma_D} g\, \boldsymbol{\omega}^- \cdot \boldsymbol{n}$$

  holds for every $\boldsymbol{\omega} \in V_h^d$.

[1] To obtain uniform stability in the limit of large p, τ_0 should also scale with p^2 [33]; we do not consider this aspect for the moderate values of p tested in this work ($p = 1$–8).

[2] The *strong-weak form* states that $\boldsymbol{q}_h$ must satisfy $\int_E \boldsymbol{q}_h \cdot \boldsymbol{\omega} = \int_E \nabla u_h \cdot \boldsymbol{\omega} + \int_{\partial E} (\widehat{u}_h - u_h)\, \boldsymbol{\omega} \cdot \boldsymbol{n}$ whereas the *weak form* states that $\boldsymbol{q}_h$ must satisfy $\int_E \boldsymbol{q}_h \cdot \boldsymbol{\omega} = -\int_E u_h \nabla \cdot \boldsymbol{\omega} + \int_{\partial E} \widehat{u}_h\, \boldsymbol{\omega} \cdot \boldsymbol{n}$. The two forms are equivalent whenever the employed quadrature scheme exactly satisfies the identity of integration by parts, which in practice is generally true for quadrilateral, prismatic, simplicial elements, etc., but is generally not true when approximate numerical quadrature schemes are used, e.g., as on implicitly defined curved elements. In the latter situation, to ensure symmetry of the final discrete Laplacian operator, it is necessary to use the strong-weak form to define $\boldsymbol{q}_h$ and the weak form to define the divergence of $\boldsymbol{q}_h$ (or vice versa) [36].

Accordingly, (2.10) is equivalent to the statement that

$$\boldsymbol{q}_h = (\nabla_h + L)u_h + J_D(g) = Gu_h + J_D(g), \tag{2.11}$$

where $G : V_h \to V_h^d$ is the *discrete gradient operator*, $G = \nabla_h + L$. The formula (2.11) is the LDG discretization of the statement $\boldsymbol{q} = \nabla u$, taking into account Dirichlet boundary data.

Next, we particularize (2.6) for the LDG method, which is the weak statement that $-\nabla \cdot \boldsymbol{q} = f$. Upon summing (2.6) over every mesh element and using the definition of the numerical flux $\widehat{\boldsymbol{q}}_h$ in (2.7), we have that, for any $v \in V_h$,

$$\begin{aligned}\sum_{E\in\mathcal{E}} \int_E \boldsymbol{q}_h \cdot \nabla v - \int_{\Gamma_0} (\{\!\{\boldsymbol{q}_h\}\!\} + \boldsymbol{\beta}[\![\boldsymbol{q}_h]\!] - \tau_0[\![u_h]\!]) \cdot [\![v]\!] - \int_{\Gamma_D} (\boldsymbol{q}_h^- \cdot \boldsymbol{n} - \tau_D u_h^-)\, v^- \\ = \int_\Omega f\, v + \tau_D \int_{\Gamma_D} g\, v^- + \int_{\Gamma_N} h\, v^-.\end{aligned} \tag{2.12}$$

Additionally, define the following operators:

- Similar to the operator J_D above, let $J_N(h) \in V_h$ be such that
$$\int_\Omega J_N(h)\, v = \int_{\Gamma_N} h\, v^-$$
for all $v \in V_h$.
- Let $E_0, E_D : V_h \to V_h$ be the operators such that, for each $u \in V_h$,
$$\int_\Omega E_0(u)\, v = \int_{\Gamma_0} [\![u]\!] \cdot [\![v]\!], \qquad \int_\Omega E_D(u)\, v = \int_{\Gamma_D} u^- v^-$$
hold for every $v \in V_h$. These operators penalize jumps in the discrete solution on interior and Dirichlet boundary faces, respectively.
- Let $a_D(g) \in V_h$ be such that
$$\int_\Omega a_D(g)\, v = \int_{\Gamma_D} g\, v^-$$
for all $v \in V_h$.

Then, using the fact that $(\boldsymbol{q}_h, \nabla_h v) + (\boldsymbol{q}_h, Lv) = (\boldsymbol{q}_h, Gv)$, (2.12) is equivalent to

$$(\boldsymbol{q}_h, Gv) + \tau_0(E_0 u_h, v) + \tau_D(E_D u_h, v) = (f + J_N(h) + \tau_D a_D(g), v), \tag{2.13}$$

or, putting aside penalty terms, $G^*\boldsymbol{q}_h = f + J_N(h)$, where $-G^*$ is the *discrete divergence operator*, the negative adjoint of the discrete gradient operator G; this is the LDG discretization of the statement that $-\nabla \cdot \boldsymbol{q} = f$, taking into account Neumann boundary data.

2.3.1. Primal formulation. To obtain the *primal formulation* of the LDG method, we combine (2.13) with (2.11) to eliminate $\boldsymbol{q}_h$ and arrive at an equation for u_h. The primal LDG formulation of (2.1) reads as follows: find $u_h \in V_h$ such that

$$a(u_h, v) = \ell(v) \tag{2.14}$$

for all $v \in V_h$, where the bilinear form $a(\cdot, \cdot)$ is given by

$$a(u_h, v) = (Gu_h, Gv) + \tau_0(E_0 u_h, v) + \tau_D(E_D u_h, v)$$

and the linear functional $\ell(\cdot)$ is given by

$$\ell(v) = (f, v) - (J_D(g), Gv) + (J_N(h), v) + \tau_D(a_D(g), v).$$

One may verify that the bilinear form $a(u, v)$ is symmetric. Discretization of the primal form (2.14) with respect to a particular basis of V_h yields a symmetric positive (semi)definite linear system of the form[3]

$$Au_h = \ell, \tag{2.15}$$

where A is the matrix form of the negative *discrete Laplacian operator*.

2.3.2. Flux formulation. An alternative, but equivalent, characterization of the LDG method is the so-called *flux formulation*, which does not eliminate the auxiliary variable $\boldsymbol{q}_h$ from the system (2.11) and (2.13) but instead retains it as a primary unknown. The flux formulation of (2.1) then reads as follows: find $(\boldsymbol{q}_h, u_h) \in V_h^d \times V_h$ such that

$$\begin{aligned} m(\boldsymbol{q}_h, \boldsymbol{\omega}) - \operatorname{grad}(u_h, \boldsymbol{\omega}) &= j(\boldsymbol{\omega}), \\ -\operatorname{div}(\boldsymbol{q}_h, v) + \tau(u_h, v) &= k(v) \end{aligned} \tag{2.16}$$

for all $(\boldsymbol{\omega}, v) \in V_h^d \times V_h$, where

$$\begin{aligned} m(\boldsymbol{q}, \boldsymbol{\omega}) &= (\boldsymbol{q}, \boldsymbol{\omega}), \\ \operatorname{grad}(u, \boldsymbol{\omega}) &= (Gu, \boldsymbol{\omega}), \\ \operatorname{div}(\boldsymbol{q}, v) &= -(\boldsymbol{q}, Gv), \\ \tau(u, v) &= \tau_0(E_0 u, v) + \tau_D(E_D u, v), \\ j(\boldsymbol{\omega}) &= (J_D(g), \boldsymbol{\omega}), \\ k(v) &= (f, v) + (J_N(h), v) + \tau_D(a_D(g), v). \end{aligned}$$

Discretization of the flux form (2.16) with respect to a particular basis of $V_h^d \times V_h$ yields a symmetric positive (semi)definite linear system of the form

$$\begin{bmatrix} M & -MG \\ -MD & MT \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_h \\ u_h \end{bmatrix} = \begin{bmatrix} j \\ k \end{bmatrix}. \tag{2.17}$$

Here, M is the block diagonal mass matrix for V_h or V_h^d (depending on context), G is the matrix form of the discrete gradient operator, $D = -M^{-1}G^\top M$ is the matrix form of the discrete divergence operator, and $T = \tau_0 E_0 + \tau_D E_D$ contains the discrete penalty terms. Since M^{-1} is also block diagonal, we can easily take the Schur complement of M in (2.17) to obtain a linear system for the unknown vector u_h,

$$Au_h = \ell, \tag{2.18}$$

where $A = M(-DG + T) = G^\top MG + MT$ and $\ell = k - G^\top j$.

The reduced linear system (2.18) is equivalent to the discrete primal formulation (2.15). However, as we will demonstrate next in section 3, the two formulations have

[3]Throughout the paper we shall frequently use the same symbol to denote (i) elements of spaces such as V_h or operators acting on such elements, and (ii) vectors of coefficients in the chosen basis or matrices acting on such vectors. The distinction should be clear from the context. Further comments are provided in section 2.3.3.

different implications for multigrid methods. In particular, applying standard operator coarsening to the discrete primal formulation results in poor multigrid performance; coarsening the discrete flux formulation (2.17) in both $\boldsymbol{q}_h$ and u_h *before* taking the Schur complement (2.18) results in optimal multigrid performance, and is equivalent to pure geometric multigrid.

2.3.3. Remarks on the choice of basis. The analysis and discussion presented in this paper is agnostic to the particular choice of basis for the piecewise polynomial space V_h. One may use a nodal basis, a modal basis, or some other choice, provided it is understood that every basis-dependent matrix (e.g., the mass matrix M) is defined consistently, relative to the chosen basis. In a numerical implementation, one should consider aspects of conditioning, accuracy, stability, sparsity, and computational complexity. For example, for low-to-moderate polynomial degree on rectangular elements, a nodal basis using Gauss–Lobatto nodes is a natural choice [26]; for very large p, a modal basis may have better conditioning or improved cost of mass matrix inversion, and thus may be more suitable. In our particular implementation, we have used a tensor-product Gauss–Lobatto nodal basis. We emphasize, however, that the presented multigrid methods and the essential conclusions drawn are not dependent on this choice.

3. Multigrid methods. We assume here that the reader has some familiarity with multigrid methods; see, for example, Briggs, Henson, and McCormick [15] for a review of their design and operation. A geometric multigrid method consists of four main ingredients: a mesh hierarchy, an interpolation operator to transfer approximate solutions from a coarse mesh onto a fine mesh, a restriction operator to formulate a coarse mesh correction problem by restricting the residual from the fine mesh, and a smoother/relaxation method. We consider these ingredients separately first, and then combine them into a multigrid V-cycle.

Multigrid methods rely on the complementarity between relaxation and interpolation. In the geometric multigrid context, a relaxation method that is effective at damping high-frequency errors, but slow to damp smooth, low-frequency ones, benefits from the action of an interpolation operator that can accurately transfer low-frequency information. By solving a correction equation for the error on a coarser grid, fine-grid low-frequency errors become coarse-grid high-frequency errors for which coarse-grid relaxation is effective. An interpolation operator then transfers this low-frequency correction to the fine grid.

In the following sections we focus our description on h-multigrid methods, wherein the mesh is coarsened geometrically at each level. However, much of our analysis carries over to p-multigrid methods, which hold the mesh fixed and instead coarsen the polynomial space by reducing p at each level. We will try to point out the distinctions between the two methods when the analogues are not immediately obvious, though we will use the notation h in our description.

3.1. Mesh hierarchy. In this work, we employ quadtrees (in 2D) and octrees (in 3D) to define the finest mesh—whether it is uniform, adaptively refined, or used as the background grid for an implicitly defined mesh (see section 4.4). The tree structure then naturally defines a hierarchy of nested meshes for use in h-multigrid, which are spatially coarsened by a factor of two in each dimension on each level. For adaptively refined meshes where the cell size is not uniform, we coarsen each element as rapidly as the tree structure permits (see, e.g., Figure 4.5).

In the context of p-multigrid methods, a mesh hierarchy is defined by applying

a specific p-coarsening strategy to the fine mesh. For example, one could coarsen p sequentially ($p \to p-1 \to p-2 \to \cdots \to 1$), by a factor of two ($p \to p/2 \to p/4 \to \cdots \to 1$), or by some user-defined sequence of p's. The first method is a common choice when low-order polynomials are used on the finest mesh, whereas the second method is better suited to high-order discretizations.

Mesh hierarchies can also be generated by combining coarsening strategies in both h and p. For example, a popular choice is to layer p-multigrid on top of h-multigrid, so that h-multigrid with a low-order polynomial degree is used as the bottom solver in the p-multigrid hierarchy.

3.2. Interpolation. The interpolation operator I_c^f transfers a piecewise polynomial function $u_c \in V_{2h}(\mathcal{E}_c)$ defined on a coarse mesh $\mathcal{E}_c$ to a piecewise polynomial function $u_f \in V_h(\mathcal{E}_f)$ on a fine mesh $\mathcal{E}_f$. (Throughout this work, subscripts or superscripts f, h and c, $2h$ shall denote objects corresponding to the fine mesh and coarse mesh, respectively.) We define the interpolation operator so that it injects the piecewise polynomial function on the coarse mesh into the fine mesh, unmodified. From the h-multigrid perspective, $u_f|_{E_f}$ on the fine element E_f is simply the polynomial $u_c|_{E_c}$ restricted to E_f, where $E_c \supset E_f$ is the corresponding coarse element in the mesh hierarchy. From the p-multigrid perspective, the lower-degree polynomial u_c can be exactly represented as a higher-degree polynomial by taking the higher-order coefficients of u_f to be zero. In either case, when regarded as an operator from $L^2(\Omega) \to L^2(\Omega)$, I_c^f is the identity operator. The operator is linear and has the property that it preserves constant functions, i.e., $u_c \equiv 1$ is mapped to $u_f \equiv 1$. This property ensures that, throughout a V-cycle, the coarse mesh discrete problems preserve the compatibility condition required in semidefinite problems having solely Neumann boundary conditions.

3.3. Restriction. We define the restriction operator $R_f^c : V_h(\mathcal{E}_f) \to V_{2h}(\mathcal{E}_c)$ as the adjoint of the interpolation operator, i.e., such that

$$(R_f^c u_f, u_c)_{\mathcal{E}_c} = (u_f, I_c^f u_c)_{\mathcal{E}_f} \tag{3.1}$$

holds for every $u_f \in V_h(\mathcal{E}_f)$ and every $u_c \in V_{2h}(\mathcal{E}_c)$. Equivalently, letting I_c^f, R_f^c, u_c, and u_f also denote matrices and vectors relative to the user-defined bases of $V_h(\mathcal{E}_f)$ and $V_{2h}(\mathcal{E}_c)$, (3.1) can be restated as

$$(R_f^c u_f)^\top M_c u_c = u_f^\top M_f I_c^f u_c,$$

where M_c and M_f are the block-diagonal mass matrices of the coarse and fine meshes, respectively. Therefore,

$$R_f^c = M_c^{-1} (I_c^f)^\top M_f. \tag{3.2}$$

Defining the restriction operator in this manner—sometimes referred to as Galerkin projection—results in several notable properties:

- One may interpret $R_f^c u_f$ as "averaging" elemental polynomials of $u_f \in V_h(\mathcal{E}_f)$ on the fine mesh to determine a coarsened piecewise-polynomial representation on the coarse mesh. The averaging is performed in a way that locally preserves the mass of u_f: indeed, since I_c^f preserves constant functions, we have that $(R_f^c u_f, 1)_{\mathcal{E}_c} = (u_f, 1)_{\mathcal{E}_f}$ for all $u_f \in V_h(\mathcal{E}_f)$.
- The adjoint method can also be viewed as an L^2 projection of $u_f \in V_h(\mathcal{E}_f)$ onto $V_{2h}(\mathcal{E}_c)$. The variational problem $\arg\min_{u_c \in V_{2h}(\mathcal{E}_c)} \|u_c - u_f\|_\Omega^2$ optimizes

the functional

$$V_{2h}(\mathcal{E}_c) \ni u_c \mapsto (u_c, u_c)_{\mathcal{E}_c} - 2(u_f, I_c^f u_c)_{\mathcal{E}_f} = u_c^\top M_c u_c - 2u_f^\top M_f I_c^f u_c$$

whose unique minimum is given by $u_c = M_c^{-1}(I_c^f)^\top M_f u_f$.

- From the preceding property, one can immediately infer that

$$R_f^c I_c^f = \mathbb{I}, \tag{3.3}$$

where $\mathbb{I}$ is the identity operator; i.e., interpolating a piecewise polynomial function from a coarse mesh onto a fine mesh and immediately restricting the result shall return the original function. The relation in (3.3) together with (3.2) also provides a method to compute the coarse-mesh mass matrix from the fine-mesh mass matrix:

$$M_c = (I_c^f)^\top M_f I_c^f. \tag{3.4}$$

Given a linear operator $A : V_h(\mathcal{E}_f) \to V_h(\mathcal{E}_f)$, one can define a coarsened operator $\mathcal{C}(A) : V_{2h}(\mathcal{E}_c) \to V_{2h}(\mathcal{E}_c)$ in a similar way, by proceeding variationally: we define $\mathcal{C}(A)$ such that

$$(\mathcal{C}(A)u_c, v_c)_{\mathcal{E}_c} = (AI_c^f u_c, I_c^f v_c)_{\mathcal{E}_f}$$

holds for all $u_c, v_c \in V_{2h}(\mathcal{E}_c)$. Viewing A and $\mathcal{C}(A)$ as matrix operators, mapping vectors in the user-defined bases of $V_h(\mathcal{E}_f)$ and $V_{2h}(\mathcal{E}_c)$,

$$\mathcal{C}(A) = M_c^{-1}(I_c^f)^\top M_f A I_c^f = R_f^c A I_c^f.$$

The last form is perhaps more commonly seen or referred to as *"RAT"* in the multigrid literature [42], where R is restriction, A is the fine-mesh operator, and T (or P) is the interpolation (or prolongation) operator; the essence of the present work is to show that directly applying RAT to the negative discrete Laplacian resulting from the primal formulation of an LDG method results in an inefficient multigrid algorithm and that, instead, applying RAT to the flux formulation, $\boldsymbol{q} = \nabla u$, $-\nabla \cdot \boldsymbol{q} = f$, leads to more efficient multigrid solvers.

3.4. Operator coarsening and pure geometric multigrid. In this section we compare a standard, purely geometric multigrid method to two multigrid schemes based on operator coarsening: (i) applying RAT to the negative discrete Laplacian matrix of the primal formulation ("primal coarsening") and (ii) applying RAT to the 2×2 block matrix of the flux formulation ("flux coarsening"). By a pure geometric method, we mean one in which each level of the hierarchy is explicitly meshed and the LDG formulation is canonically applied to each level, with the above restriction and interpolation operators transferring residual and correction vectors (in a V-cycle) between levels. Our motivation here concerns an h-multigrid method; however, much of the following discussion has direct analogy with p-multigrid methods. In addition, in the context of DG methods requiring penalty parameters, a design choice can be made as to how the value of the penalty parameter is chosen at each level of the hierarchy. In this work we consider the natural choice in which every level of the hierarchy inherits the same value as the finest mesh. With this in mind, we discuss interaction between a pair of levels: suppose $\mathcal{E}_f$ is the mesh of a fine level and $\mathcal{E}_c$ is the mesh of the next-coarsest level.

3.4.1. Primal coarsening. Recall the primal form of the negative discrete Laplacian operator of an LDG method: as a matrix mapping the coefficient vectors in the basis of $V_h(\mathcal{E}_f)$ into the basis of $V_h(\mathcal{E}_f)$, i.e., premultiplying (2.18) by M^{-1},

$$A = -DG + \tau_0 E_0 + \tau_D E_D,$$

where $G = \nabla_h + L$ is the discrete gradient operator and $D = -M^{-1}G^\top M$ is the discrete divergence operator. To discuss the application of RAT to A and how it relates to a geometric multigrid implementation, we consider the individual terms making up A.

- First, we note that the broken gradient operator satisfies the RAT property, i.e., $\mathcal{C}(\nabla_h) = \nabla_{2h}$. Computing the piecewise gradient on a coarse mesh and interpolating the result to the fine mesh is the same as computing the piecewise gradient of the interpolant, i.e., $I_c^f \nabla_{2h} u_c = \nabla_h I_c^f u_c$ for all $u_c \in V_{2h}(\mathcal{E}_c)$; consequently, $\mathcal{C}(\nabla_h) = R_f^c \nabla_h I_c^f = R_f^c I_c^f \nabla_{2h} = \nabla_{2h}$ by (3.3).
- The lifting operator also satisfies the RAT property, i.e., $\mathcal{C}(L_f) = L_c$. This is perhaps not immediately obvious, since source terms on a coarse mesh face will lift into the corresponding large coarse element, whereas the corresponding source terms on the fine mesh faces lift only into the smaller elements touching that face; however, the restriction of the result on the set of smaller elements agrees with the result of L_c. To see this, apply the variational formulation of $\mathcal{C}(\cdot)$ to observe that

$$\begin{aligned}
&(\mathcal{C}(L_f)u_c, \boldsymbol{v}_c)_{\mathcal{E}_c} \\
&\quad = (L_f I_c^f u_c, I_c^f \boldsymbol{v}_c)_{\mathcal{E}_f} \\
&\quad = -\int_{\Gamma_{0,f}} [\![I_c^f u_c]\!] \cdot \big(\{\!\{ I_c^f \boldsymbol{v}_c \}\!\} + \boldsymbol{\beta} [\![I_c^f \boldsymbol{v}_c]\!] \big) - \int_{\Gamma_{D,f}} (I_c^f u_c)^- (I_c^f \boldsymbol{v}_c)^- \cdot \boldsymbol{n} \\
&\quad = -\int_{\Gamma_{0,c}} [\![u_c]\!] \cdot (\{\!\{ \boldsymbol{v}_c \}\!\} + \boldsymbol{\beta} [\![\boldsymbol{v}_c]\!]) - \int_{\Gamma_{D,c}} u_c^- \boldsymbol{v}_c^- \cdot \boldsymbol{n} \\
&\quad = (L_c u_c, \boldsymbol{v}_c)_{\mathcal{E}_c}
\end{aligned}$$

  holds for all $u_c \in V_{2h}(\mathcal{E}_c)$ and $\boldsymbol{v}_c \in V_{2h}^d(\mathcal{E}_c)$. Here, $\Gamma_{0,f}$ and $\Gamma_{0,c}$ denote the union of interior faces of the fine and coarse meshes, respectively, and similarly for $\Gamma_{D,f}$ and $\Gamma_{D,c}$. The third equality holds because the interpolation operator introduces no nonzero jumps on the set of new fine mesh faces, i.e., on $\Gamma_{0,f} \setminus \Gamma_{0,c}$ and $\Gamma_{D,f} \setminus \Gamma_{D,c}$. (The preceding assumes that fine mesh faces inherit the same $\boldsymbol{\beta}$ value as coarse mesh faces; in particular, this is true for the one-sided LDG scheme in which $\boldsymbol{\beta} = \pm\frac{1}{2}\boldsymbol{n}$.)
- It immediately follows from the preceding two properties that $\mathcal{C}(G_f) = G_c$. Moreover,

$$\begin{aligned}
\mathcal{C}(D_f) = R_f^c D_f I_c^f &= \big(M_c^{-1}(I_c^f)^\top M_f\big)\big(-M_f^{-1} G_f^\top M_f\big) I_c^f \\
&= -M_c^{-1}\big((I_c^f)^\top G_f^\top M_f I_c^f M_c^{-1}\big) M_c = -M_c^{-1}\big(\mathcal{C}(G_f)\big)^\top M_c \\
&= -M_c^{-1} G_c^\top M_c = D_c.
\end{aligned}$$

- It is straightforward to show that the penalty operators also satisfy the RAT property, i.e., $\mathcal{C}(E_{0,f}) = E_{0,c}$ and $\mathcal{C}(E_{D,f}) = E_{D,c}$. As in the case of the lifting operator, this property derives from the fact the interpolation operator

does not introduce jumps on fine mesh faces that do not overlap with coarse mesh faces.

Despite these consistencies, the negative discrete Laplacian does not satisfy the *RAT* property—the application of *RAT* to the fine-mesh negative discrete Laplacian A_f does not yield the coarse-mesh operator A_c obtained from pure geometric multigrid. Using the properties derived above,

$$\begin{aligned} A_c &= -D_c G_c + \tau_0 E_{0,c} + \tau_D E_{D,c} \\ &= -\mathcal{C}(D_f)\mathcal{C}(G_f) + \tau_0 \mathcal{C}(E_{0,f}) + \tau_D \mathcal{C}(E_{D,f}) \end{aligned}$$

which differs from the direct coarsening of A_f,

$$\mathcal{C}(A_f) = -\mathcal{C}(D_f G_f) + \tau_0 \mathcal{C}(E_{0,f}) + \tau_D \mathcal{C}(E_{D,f}) \neq A_c,$$

since, in general, $\mathcal{C}(D_f G_f) \neq \mathcal{C}(D_f)\mathcal{C}(G_f)$. Informally, $\mathcal{C}(D_f G_f)u_c$ interpolates a function $u_c \in V_{2h}(\mathcal{E}_c)$ onto the fine mesh $\mathcal{E}_f$, computes the gradient as a function in $V_h^d(\mathcal{E}_f)$, computes the divergence as a function in $V_h(\mathcal{E}_f)$, and projects the result back to the coarse mesh $\mathcal{E}_c$. On the other hand, $\mathcal{C}(D_f)\mathcal{C}(G_f)u_c$ projects the computed fine-mesh gradient onto the coarse mesh and then immediately interpolates the result in order to compute the discrete divergence on the fine mesh, before projecting the final result back to the coarse mesh. That is,

$$\mathcal{C}(D_f)\mathcal{C}(G_f) = \mathcal{C}(D_f I_c^f R_f^c G_f) \neq \mathcal{C}(D_f G_f),$$

since $I_c^f R_f^c \neq \mathbb{I}$.

3.4.2. Flux coarsening. The coarse operator A_c obtained from pure geometric multigrid may be viewed as applying *RAT* to the equations $\boldsymbol{q} = \nabla u$ and $-\nabla \cdot \boldsymbol{q} = f$ separately. The flux formulation of LDG, (2.17), naturally displays this coarsening strategy. To show this, note that we can write the flux formulation with input and output in the user-defined basis by premultiplying (2.17) by the inverse mass matrix to obtain

$$\begin{bmatrix} I & -G \\ -D & T \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_h \\ u_h \end{bmatrix} = \begin{bmatrix} M^{-1} j \\ M^{-1} k \end{bmatrix}. \tag{3.5}$$

Applying *RAT* in a block fashion to the flux formulation (3.5) then yields the discrete operator

$$\begin{bmatrix} R_f^c & 0 \\ 0 & R_f^c \end{bmatrix} \begin{bmatrix} I & -G_f \\ -D_f & T_f \end{bmatrix} \begin{bmatrix} I_c^f & 0 \\ 0 & I_c^f \end{bmatrix} = \begin{bmatrix} I & -G_c \\ -D_c & T_c \end{bmatrix}. \tag{3.6}$$

Taking the Schur complement of the right-hand side of (3.6), we obtain

$$\begin{aligned} A_c &= -D_c G_c + T_c \\ &= -\mathcal{C}(D_f)\mathcal{C}(G_f) + \tau_0 \mathcal{C}(E_{0,f}) + \tau_D \mathcal{C}(E_{D,f}), \end{aligned} \tag{3.7}$$

which is exactly the coarse operator from pure geometric multigrid. Thus, applying operator coarsening to the flux formulation of LDG, which is equivalent to separately coarsening the equations $\boldsymbol{q} = \nabla u$ and $-\nabla \cdot \boldsymbol{q} = f$, is the same as pure geometric multigrid.

Figure 3.1 depicts the three types of coarsening that can be performed, given a hierarchy of meshes. In the left column, pure geometric multigrid defines the coarse operators directly from the coarse meshes; in the center column, primal coarsening applies RAT to the discrete Laplacian operator; and in the right column, flux coarsening applies RAT separately to the discrete divergence and gradient operators. In the above, we have shown the equivalence of the left and right columns. An implementation of constructing the operator hierarchy using flux coarsening is outlined in Algorithm 3.1.

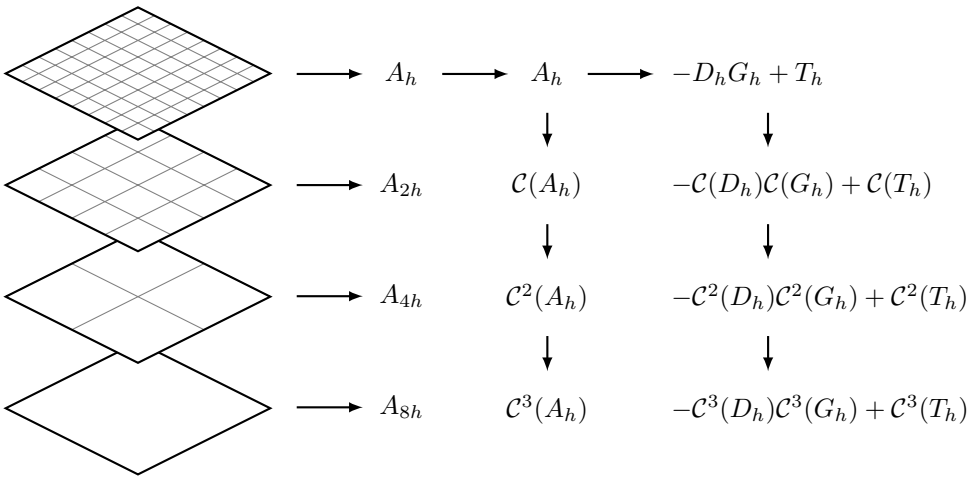

FIG. 3.1. *Three coarsening methods can be used to generate a hierarchy of operators for multigrid: (left column) pure geometric multigrid, where the coarse operators are defined directly from the corresponding coarse meshes, (center column) primal coarsening, where the coarse operators are defined by applying RAT to the fine-mesh discrete Laplacian, and (right column) flux coarsening, where the coarse operators are defined by applying RAT to the fine-mesh discrete divergence, discrete gradient, and discrete penalty operators, and recombining the results.*

3.4.3. Benefits of operator coarsening. It can be useful to define coarse operators directly from fine operators (e.g., by using RAT) rather than via discretizations computed directly from coarse meshes. Since coarse mass matrices can be computed automatically according to (3.4), quadrature schemes do not need to be computed for coarse elements—instead, fine-mesh quadrature rules are coarsened automatically via (3.4). Similarly, coarse lifting matrices are not explicitly needed since their contribution to the discrete gradient is automatically computed via $G_c = \mathcal{C}(G_f)$, and so quadrature rules for coarse faces also do not need to be defined. For implicitly defined meshes, such as the ones shown in section 4.4, computing coarse quadrature rules can be computationally intricate or taxing; the fact that operator coarsening obviates the need for this is a substantial benefit. Additionally, operator coarsening can be efficiently implemented using basic linear algebra operations, e.g., block-sparse matrix multiplication, for which highly optimized and parallelized libraries exist; in contrast, computing discretizations directly from coarse meshes relies heavily on the efficiency of one's own code. It is worth noting that the complexity of constructing the operator hierarchy in Algorithm 3.1 is the same as the complexity of the multigrid V-cycle in Algorithm 3.2 (i.e., $\mathcal{O}(N)$ for N elements); as an approximate indication, in practice the former takes the same computing time as about three to four applications of a V-cycle.

Algorithm 3.1. Construction of coarse operators, Build$(\mathcal{E}_f, M_f, G_f, T_f)$.

Input: Fine-mesh operators M_f, G_f, T_f
Output: List of coarse operators
$\boldsymbol{A} := \{\}$
$A_f := G_f^\top M_f G_f + T_f$
if $\mathcal{E}_f$ is not the coarsest mesh **then**
  $M_c := (I_c^f)^\top M_f I_c^f$
  $G_c := M_c^{-1}(I_c^f)^\top M_f G_f I_c^f$
  $T_c := (I_c^f)^\top T_f I_c^f$
  $\boldsymbol{A} :=$ Build$(\mathcal{E}_c, M_c, G_c, T_c)$
return $\{A_f, \boldsymbol{A}\}$

Algorithm 3.2. Multigrid V-cycle $V(\mathcal{E}_f, x_f, b_f)$ on mesh $\mathcal{E}_f$ with ν pre- and post-smoothing steps.

if $\mathcal{E}_f$ is the bottom level **then**
  Solve $A_f x_f = b_f$ directly
else
  Relax ν times
  $r_c := (I_c^f)^\top (b_f - A_f x_f)$
  $x_c := V(\mathcal{E}_c, 0, r_c)$
  $x_f := x_f + I_c^f x_c$
  Relax ν times
return x_f

3.4.4. Relation to other DG methods. Although we have focused on the LDG method in our discussion, we expect that other DG methods may require similar care in coarsening fine-grid operators such that they are consistent with a pure geometric multigrid method. For instance, methods for which the numerical flux $\widehat{\boldsymbol{q}}_h$ depends on the discrete gradient of u_h—such as the BR1 [9] or Brezzi [14] methods—may need similar treatment, as the contribution from the lifting operator L must be coarsened separately.

Other methods, such as the symmetric interior penalty (SIP) method [21, 6], do not require the discrete divergence and discrete gradient operators to be coarsened separately. To demonstrate this for SIP, we start with its corresponding bilinear form for a pure Neumann problem: find $u \in V_h$ such that $a(u, v) = l(v)$ for all $v \in V_h$, where

$$a(u,v) = (\nabla_h u, \nabla_h v) - \int_{\Gamma_0} (\{\!\{\nabla_h u\}\!\} \cdot [\![v]\!] + [\![u]\!] \cdot \{\!\{\nabla_h v\}\!\}) + \tau \int_{\Gamma_0} [\![u]\!] \cdot [\![v]\!]$$

and

$$l(v) = (f, v) + \int_{\Gamma_N} h\, v^-,$$

with τ scaling inversely to the element size. Now consider a pure geometric multigrid method. Let $u_c, v_c \in V_{2h}(\mathcal{E}_c)$. Then

$$\begin{aligned} a_c(u_c, v_c) &= (\nabla_{2h} u_c, \nabla_{2h} v_c) - \int_{\Gamma_{0,c}} (\{\!\{\nabla_{2h} u_c\}\!\} \cdot [\![v_c]\!] + [\![u_c]\!] \cdot \{\!\{\nabla_{2h} v_c\}\!\}) \\ &\quad + \tau \int_{\Gamma_{0,c}} [\![u_c]\!] \cdot [\![v_c]\!] \\ &= (\nabla_h I_c^f u_c, \nabla_h I_c^f v_c) - \int_{\Gamma_{0,f}} (\{\!\{\nabla_h I_c^f u_c\}\!\} \cdot [\![I_c^f v_c]\!] + [\![I_c^f u_c]\!] \cdot \{\!\{\nabla_h I_c^f v_c\}\!\}) \\ &\quad + \tau \int_{\Gamma_{0,f}} [\![I_c^f u_c]\!] \cdot [\![I_c^f v_c]\!] \\ &= a_f(I_c^f u_c, I_c^f v_c), \end{aligned}$$

where equality holds between the first and second lines because I_c^f does not introduce nonzero jumps on new mesh faces. Therefore, as matrices (mapping vectors in the

user-defined basis to vectors in the same basis),

$$u_c^\top M_c A_c v_c = (I_c^f u_c)^\top M_f A_f (I_c^f v_c).$$

Assuming the quadratic form is nondegenerate (which is true since A_c and A_f are symmetric positive definite, ignoring the trivial kernel), this implies

$$A_c = M_c^{-1}(I_c^f)^\top M_f A_f I_c^f = R_f^c A_f I_c^f.$$

This is RAT applied to A_f, and so applying pure geometric multigrid to SIP is the same as recursively applying standard (primal) operator coarsening to A_f.

3.5. Multigrid preconditioned conjugate gradient. Recall that a geometric multigrid method utilizes a combination of relaxation/smoothing together with interpolated approximate solutions of coarsened problems. In the present case, the coarsened problem solves for the correction in a residual equation for the same elliptic problem except on a coarser mesh. In particular, it is important to note that, instead of solving $-\Delta_h u = f$, where Δ_h is the discrete Laplacian in the chosen basis, one instead solves $-M\Delta_h u = Mf$, as the latter system is symmetric positive (semi)definite. Thus, to appropriately define the coarse mesh problem, one may: (i) calculate the residual of the fine mesh linear system $A_f x_f = b_f$, (ii) multiply the residual by M_f^{-1} to correctly determine the residual as a piecewise polynomial function, (iii) restrict the residual to the coarse mesh, and then (iv) multiply this residual by M_c of the coarse mesh. Thus, the coarse mesh problem consists of (approximately) solving for x_c such that

$$A_c x_c = M_c(R_f^c[M_f^{-1}(b_f - A_f x_f)]),$$

which, according to the derived restriction operator (3.2), conveniently simplifies to

$$A_c x_c = (I_c^f)^\top (b_f - A_f x_f),$$

and so it is unnecessary to multiply by mass matrices in the implementation of the multigrid method; instead, one can simply apply the transpose of the interpolation matrix. With this consideration in mind, the design of a multigrid V-cycle is relatively straightforward and is outlined in Algorithm 3.2.

The V-cycle is designed to preserve the symmetric positive (semi)definite property of the discrete problem, making it suitable for preconditioning the conjugate gradient method. To that end, the relaxation sweeps are performed in a symmetric fashion; for order-dependent relaxation methods such as Gauss–Seidel, the first set of relaxation sweeps uses a given ordering of the unknowns and the second set uses the reverse of that ordering.[4] A multigrid preconditioned conjugate gradient method (MGPCG) [41] combines the advantages of both solvers: the multigrid preconditioner is effective in the interior of the domain where the elliptic behavior of the matrix dominates, while the conjugate gradient method effectively treats the remaining eigenmodes, which in turn are largely associated with the (weak) imposition of the boundary conditions (and in the case of multiphase elliptic interface problems, jump conditions on internal interfaces) [40, 36]. We use a single multigrid V-cycle as a preconditioner in the conjugate gradient method.

[4]If a relaxation scheme is used that is itself symmetric, then there is no need to reverse the ordering of unknowns between pre- and post-smoothing steps, as the V-cycle will automatically preserve symmetry.

4. Numerical results. In this section, we present numerical experiments to assess the efficacy of flux coarsening for LDG discretizations of elliptic PDEs. As the smoother/relaxation method, we use a block Gauss–Seidel smoother with $\nu = 3$ pre- and post-smoothing steps[5] in the V-cycle. We initially set the interior penalty parameter to $\tau_0 = 0.01/h$; additional analysis of the influence of penalty parameters is given in section 4.2. We measure multigrid performance via the average convergence factor

$$\rho = \exp\left(\frac{1}{N}\log\frac{\|e_N\|_2}{\|e_0\|_2}\right), \tag{4.1}$$

where N is the number of iterations required to reduce the relative error by a factor of 10^{-10} and e_i is the error at iteration i of either standalone multigrid (i.e., i many V-cycles) or MGPCG (i.e., the ith iteration of CG preconditioned by a single V-cycle). In effect, ρ measures the average slope of e_i on a log-linear graph. Convergence is measured using a right-hand side of $f = 0$ with a random nonzero initial guess for u. Convergence results are presented in the following graphs with ρ as a function of element size h, polynomial degree p, etc. The same data is presented in tabular form in the supplementary material attached to this paper.

4.1. Uniform Cartesian grids. We start by solving (2.1) with homogeneous Neumann boundary conditions on the domain $\Omega = [0,1]^d$ using a uniform Cartesian grid of size $n \times n$ (for $d = 2$) or $n \times n \times n$ (for $d = 3$) with cell size $h = 1/n$. We build an h-multigrid hierarchy based on uniform grid refinement by applying both primal and flux coarsening to the discretized LDG system, and solve using both standalone V-cycles and MGPCG.

Figures 4.1 and 4.2 show the average convergence factor versus n for polynomial orders $1 \le p \le 5$ in 2D and 3D, respectively. In both cases, the multigrid scheme built using flux coarsening exhibits nearly h-independent convergence factors of $\rho \approx 0.1$, whereas the scheme based on primal coarsening exhibits poor performance that degrades as $h \to 0$.

Similar results hold for p-multigrid on uniform Cartesian grids. We generate a p-multigrid hierarchy by successively halving the polynomial order (i.e., $p \to p/2 \to p/4 \to \cdots \to 1$) and applying both primal and flux coarsening to the discretized LDG system. Figure 4.3 shows convergence factor versus p for grid sizes $4 \le n \le 512$ in 2D and 3D. Again, convergence appears to be independent of p (at least up to $p = 8$) when flux coarsening is used with p-multigrid, whereas performance degrades with increasing p for primal coarsening.

4.2. On the effect of penalty parameters on multigrid performance. Figure 4.4 (left) shows a study of the impact the penalty parameter $\tau_0 = \widetilde{\tau}_0/h$ has on multigrid convergence for a Poisson problem on a uniform $n \times n$ mesh with periodic boundary conditions and $p = 2$. Smaller values of $\widetilde{\tau}_0$ yield better convergence factors; for $\widetilde{\tau}_0 > 10$, multigrid performance begins to degrade as the mesh is refined. For the remainder of our tests, we set $\widetilde{\tau}_0 = 0.01$ so that $\tau_0 = 0.01/h$.

In our tests, the imposition or combination of Dirichlet, Neumann, or periodic boundary conditions does not affect the conclusions made in this work. However, for

[5] As is typical in multigrid methods, increasing the number of pre- and post-smoothing steps can increase the speed of convergence, i.e., decrease ρ as measured by (4.1); however, doing so comes at the cost of a more expensive V-cycle and, therefore, may be less efficient. We observed that $\nu = 3$ gave the best computational efficiency in our numerical experiments in terms of reducing the error by a fixed factor.

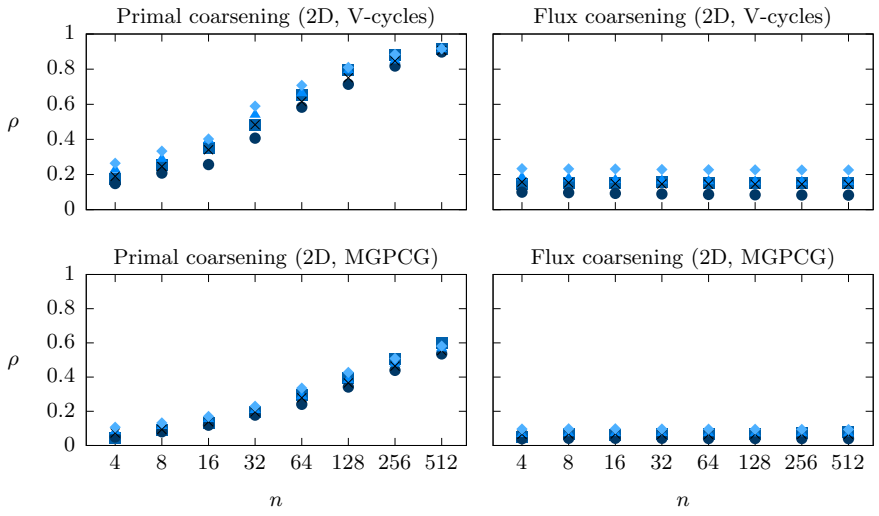

FIG. 4.1. *h-multigrid convergence factors for primal coarsening* ($A_c = \mathcal{C}(A_f)$) *and flux coarsening* ($A_c = -\mathcal{C}(D_f)\mathcal{C}(G_f) + \mathcal{C}(T_f)$) *applied to the LDG discretization of Poisson's equation on a uniform* $n \times n$ *Cartesian grid as* $h \to 0$. *The top row results are computed using V-cycles whereas the bottom row results are computed using MGPCG. The plot markers indicate different polynomial orders:* ■, ●, ▲, ×, *and* ◆ *denote* $p = 1$, 2, 3, 4, *and* 5, *respectively.*

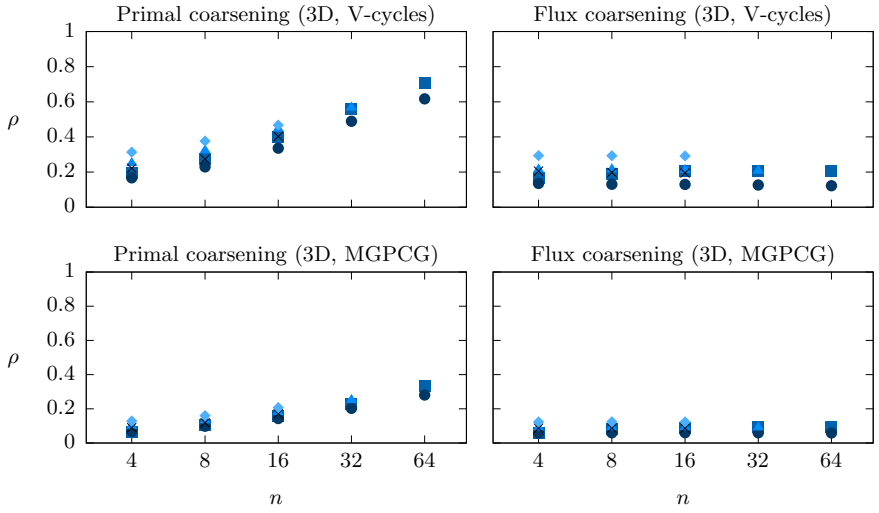

FIG. 4.2. *h-multigrid convergence factors for primal coarsening* ($A_c = \mathcal{C}(A_f)$) *and flux coarsening* ($A_c = -\mathcal{C}(D_f)\mathcal{C}(G_f) + \mathcal{C}(T_f)$) *applied to the LDG discretization of Poisson's equation on a uniform* $n \times n \times n$ *Cartesian grid as* $h \to 0$. *The top row results are computed using V-cycles whereas bottom row results are computed using MGPCG. The plot markers indicate different polynomial orders:* ■, ●, ▲, ×, *and* ◆ *denote* $p = 1$, 2, 3, 4, *and* 5, *respectively. (Omitted data points correspond to simulations whose memory requirements approximately exceed* 120 GB.*)*

problems with Dirichlet boundary conditions the choice of Dirichlet penalty parameter τ_D can impact multigrid efficiency. Informally, Dirichlet boundary conditions are enforced in a DG method only weakly and the smoothing/relaxation method of a V-cycle can only effectively enforce the boundary condition if the associated penalty parameter is sufficiently strong. Figure 4.4 (right) shows a study of the impact that $\tau_D = \widetilde{\tau}_D/h$ has on multigrid performance for a homogeneous Dirichlet problem on

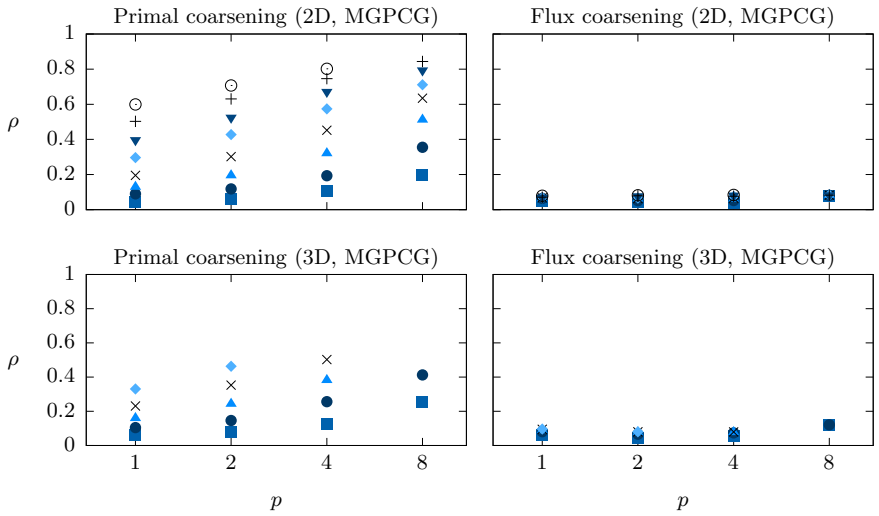

FIG. 4.3. *p-multigrid convergence factors for primal coarsening* ($A_c = \mathcal{C}(A_f)$) *and flux coarsening* ($A_c = -\mathcal{C}(D_f)\mathcal{C}(G_f) + \mathcal{C}(T_f)$) *applied to the LDG discretization of Poisson's equation on uniform grids. The p-multigrid hierarchy is generated by successively halving the polynomial order (i.e.,* $p \to p/2 \to p/4 \to \cdots \to 1$*). The plot markers indicate different grid sizes:* ■, ●, ▲, ×, ◆, ▼, +, *and* ○ *denote* $n = 4$, 8, 16, 32, 64, 128, 256, *and* 512, *respectively. (Omitted data points correspond to simulations whose memory requirements approximately exceed* 120 GB.*)*

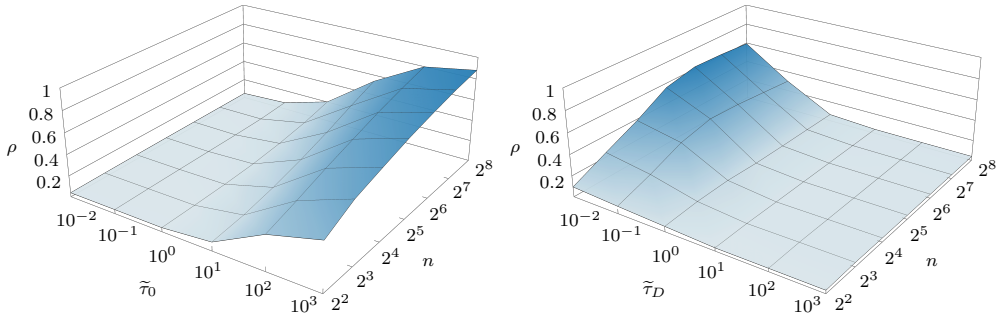

FIG. 4.4. *Effects of penalty parameter* $\tau_0 = \widetilde{\tau}_0/h$ *(left) and Dirichlet penalty parameter* $\tau_D = \widetilde{\tau}_D/h$ *(right) on multigrid convergence factor* ρ *for an* $n \times n$ *Cartesian mesh with* $p = 2$*, where* $h = 1/n$*; see section* 4.2.

a uniform $n \times n$ mesh. For $\widetilde{\tau}_D < 1$, the average convergence factor ρ degrades as n increases; a good choice in this case appears to be $10 < \widetilde{\tau}_D < 100$.

Ultimately, the proper choice of both τ_0 and τ_D is application-dependent and concerns not only multigrid performance but also discretization accuracy and effects of penalty stabilization on the conditioning of the linear systems.

4.3. Adaptive mesh refinement. Next, we solve the Neumann problem (2.1) on an adaptively refined Cartesian mesh, where refinement is performed according to some prescribed spatially-varying function. We implement adaptivity using a quadtree (in 2D) or octree (in 3D), which naturally defines a geometric hierarchy of meshes. An example hierarchy is shown in Figure 4.5. Note that some elements at various levels of the hierarchy have four neighbors on a single side, so that large elements and small elements may share part of a face.

Figure 4.6 shows the average convergence factor versus $1/h_{\min}$ for polynomial

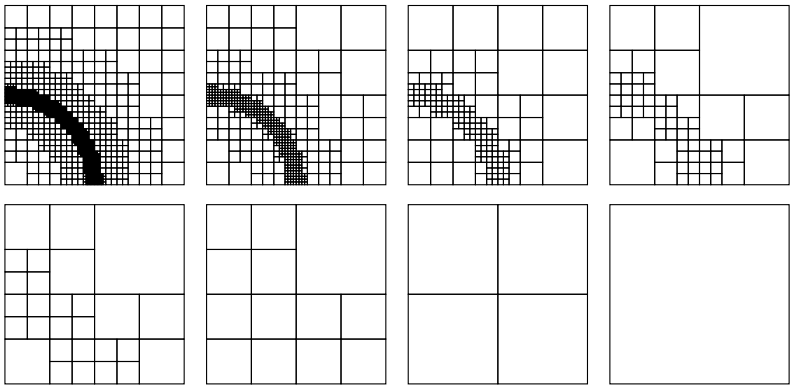

FIG. 4.5. *An example of a geometric multigrid hierarchy inherited from an adaptively refined quadtree with rapid coarsening. The finest level depicted has smallest cell size equal to* $h_{min} = 1/128$.

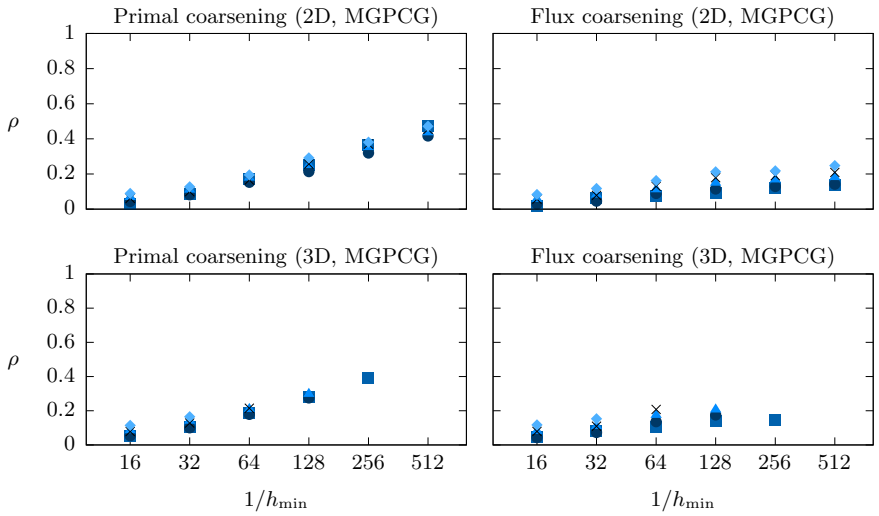

FIG. 4.6. *h-multigrid convergence factors for primal* ($A_c = \mathcal{C}(A_f)$) *and flux* ($A_c = -\mathcal{C}(D_f)\mathcal{C}(G_f) + \mathcal{C}(T_f)$) *coarsening applied to the LDG discretization of Poisson's equation on an adaptively refined grid in 2D and 3D. The plot markers indicate different polynomial orders:* ■ , ● , ▲ , × , *and* ◆ *denote* $p = 1$, 2, 3, 4, *and* 5, *respectively. (Omitted data points correspond to simulations whose memory requirements approximately exceed* 120 GB.*)*

orders $1 \le p \le 5$, where $h_{\min}$ is the size of the smallest element in the mesh. In both 2D and 3D, the multigrid method based on primal coarsening exhibits performance that degrades as $h_{\min} \to 0$, whereas the method based on flux coarsening yields good performance that is nearly independent of $h_{\min}$ for all p considered.

4.4. Implicitly defined meshes and elliptic interface problems. Our last two examples are designed to exemplify the benefits of operator coarsening by considering cases in which a pure geometric multigrid method would be intricate or difficult to implement, such as on nontrivial domains with complex geometry or for elliptic interface problems in which the interface has extreme geometry. The first example consists of a curved domain containing holes and thin pieces, and the second example is a multiphase elliptic interface problem with small circles, filaments, and cusps in the interface geometry. In both cases, we make use of a recently developed framework

for computing high-order accurate multiphase multiphysics using implicitly defined meshes [36, 37]. The framework shares some aspects with cut-cell techniques wherein a level set function defining the domain geometry or internal interfaces is used to cut through the cells of a background quadtree or octree; tiny cut cells are then merged with neighboring cells to create a mesh in which the shapes of interfacial elements are defined implicitly by the level set function. Quadrature rules for curved elements and nontrivial mesh faces are then computed using high-order accurate schemes for computing integrals on implicitly defined domains restricted to hyperrectangles [35]; these quadrature schemes are then used in the DG weak formulation, e.g., for computing mass matrices and the lifting operator L on the finest-level mesh.

In both examples, we consider an elliptic PDE problem with Dirichlet boundary conditions. The Dirichlet penalty parameter is chosen to scale inversely with h, the typical element size on the finest mesh, such that $\tau_D = 100/h$; the value 100 was determined empirically as being approximately the smallest possible while giving good multigrid performance. To measure the convergence rate, we apply the MGPCG method to a homogeneous problem with random nonzero initial condition and measure the average convergence rate using (4.1).

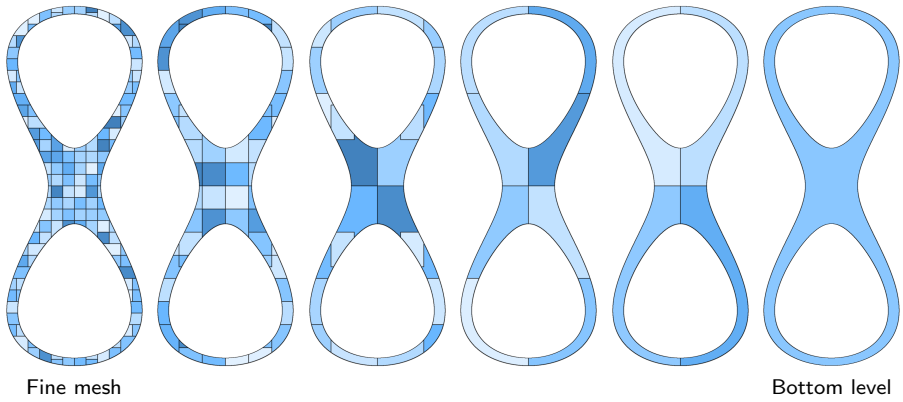

FIG. 4.7. *A single-phase test case applied to a curved domain using implicitly defined meshes. Depicted is the implied h-multigrid hierarchy (elements are randomly colored). The coarse meshes are not explicitly constructed in our operator coarsening approach; instead, the discrete gradient and penalty operators and mass matrices are constructed top-down at each level (see Algorithm* 3.1*). In particular, note that the bottom level of the hierarchy consists of a mesh with a single element containing two holes. (Figure is in color online.)*

The first example of a single-phase Poisson problem on a curved domain is illustrated in Figure 4.7 and consists of a figure-eight domain with two holes surrounded by thin segments. It is important to note that the illustrated mesh hierarchy is implicitly formed by our operator coarsening scheme in Algorithm 3.1 and it is only the finest-level mesh which is built. On coarser levels, the elements are agglomerated according to the coarsening of the background quadtree; in particular, we note that the bottom level consists of a single element containing two holes. Using the operator coarsening strategy, there is no need to compute quadrature rules for the coarse levels of the mesh hierarchy—the quadrature rules from the fine mesh are effectively coarsened automatically. The results for solving the Dirichlet problem (2.1) using flux coarsening[6]

[6]Results using primal coarsening are similar to previous examples that use primal coarsening—poor multigrid performance is observed that degrades with mesh size—and have been omitted for brevity.

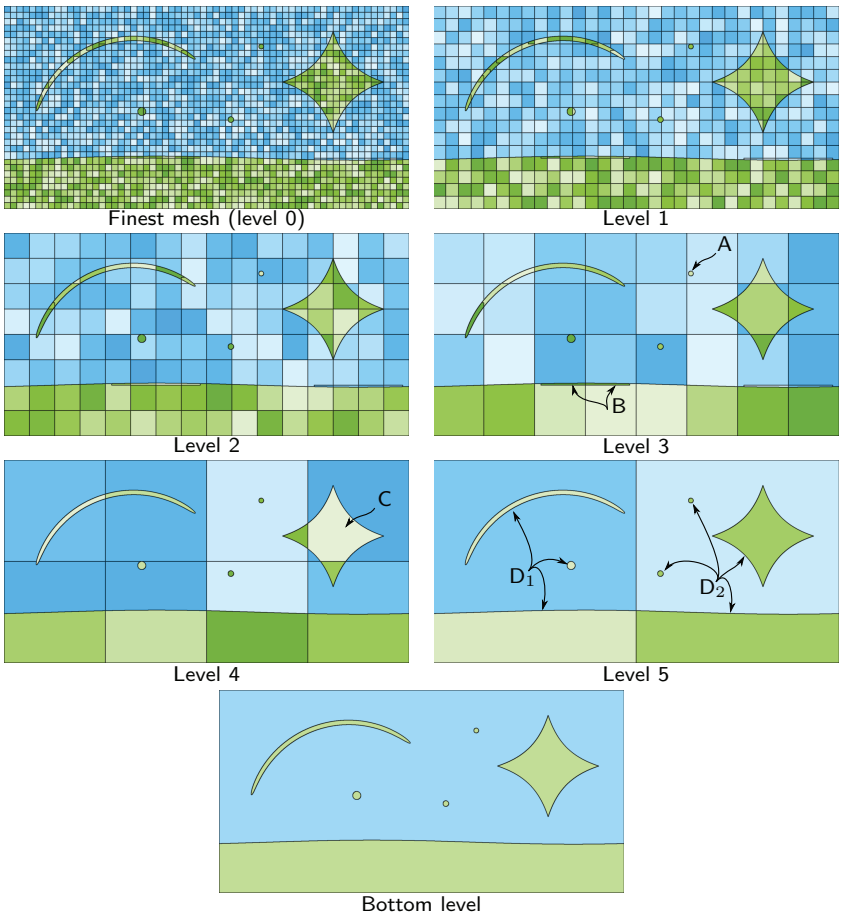

FIG. 4.8. *A test case involving a multiphase elliptic interface problem. Depicted is the implied h-multigrid hierarchy, wherein the elements are randomly colored a shade of green or blue for phase one or two, respectively. Similar to Figure* 4.7*, only the finest mesh is explicitly constructed; on coarser levels, the discrete gradient and penalty operators and mass matrices are constructed top-down at each level (see Algorithm* 3.1*). Note that the bottom level of the hierarchy consists of a mesh of just two elements—the blue element has one component with five holes, whereas the green element consists of six connected components. See the discussion following* (4.2) *for a description of the noted features A, B, C, D_1, and D_2.*

and h-multigrid on the curved domain of Figure 4.7 are shown in Figure 4.9 (left); we observe good multigrid convergence factors of $\rho \approx 0.05$–0.2, nearly independent of grid size, for $1 \le p \le 5$.

The second example considers a two-phase elliptic interface problem in a rectangular domain illustrated in Figure 4.8. The corresponding PDE consists of solving

$$\begin{aligned} -\nabla \cdot (\mu_i \nabla u) &= f \quad \text{in } \Omega_i, & [\![u]\!] &= g_\Gamma \quad \text{on } \Gamma, \\ u &= g_\partial \quad \text{on } \partial\Omega, & [\![\mu \nabla u \cdot \boldsymbol{n}]\!] &= h_\Gamma \quad \text{on } \Gamma, \end{aligned} \tag{4.2}$$

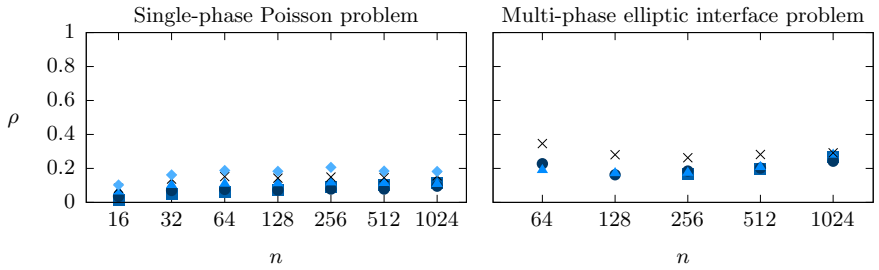

FIG. 4.9. *h-multigrid convergence factors for flux coarsening and MGPCG applied to the LDG discretization of (left) the single-phase Poisson problem on the curved domain shown in Figure* 4.7 *and (right) the multiphase elliptic interface test problem in* (4.2) *on the domain shown in Figure* 4.8. *The grid size n is the number of cells of the background Cartesian grid required to cover the longest extent of the domain. The plot markers indicate different polynomial orders:* ■ , ● , ▲ , × , *and* ◆ *denote* $p = 1$, 2, 3, 4, *and* 5, *respectively.*

where Γ is the interface between phases Ω_1 (green region, with ellipticity coefficient[7] $\mu_1 = 1$) and Ω_2 (blue region, with $\mu_2 = 4$), and $[\![\cdot]\!]$ denotes the interfacial jump in the indicated quantity. In this test case, the geometry of the interface has been designed to be challenging—the crescent shape is long and thin; there are three isolated, small circles; and the star shape has sharp cusp-like corners. Once more we note that the illustrated hierarchy in Figure 4.8 is implicitly formed by the operator coarsening strategy, and only the finest-level mesh is actually built. However, we note that the agglomeration strategy for this multiphase problem is slightly different from the previous test examples—here, elements are only agglomerated with elements belonging to the same phase. Thus, the interface remains sharp throughout all levels of the hierarchy, and this can dramatically improve the performance of multigrid methods for elliptic interface problems, especially when using high-order accurate techniques [36]. Owing to this agglomeration strategy, coarse mesh levels can have intricate element shapes. Some example features are noted in Figure 4.8—A indicates a tiny green-phase element surrounded by a large blue-phase element; B indicates two sliver elements; C indicates an element whose cusp-like corners would make it rather difficult to apply a black-box quadrature scheme if one were to explicitly build the coarse-level mesh; and $\mathsf{D}_{1,2}$ show two green-phase elements, each with multiple connected components (three for D_1 and four for D_2), which is perhaps rather unusual for a finite element method. Another aspect which motivated the present work on operator coarsening is that it would be nearly impossible to directly apply the cell merging algorithms underlying implicitly defined meshes [36] to these coarse levels. Although elements with extreme shapes like these (especially tiny elements next to large elements) can traditionally be of concern for numerical discretization of PDEs, according to our tests they pose no problem when present in the coarse levels of a multigrid solver. Results for solving the homogeneous version of (4.2) (f, g_∂, g_Γ, and h_Γ all zero) with a random nonzero initial guess using flux operator coarsening are shown in Figure 4.9 (right). In this case of an implicitly defined mesh for which different cell merging decisions take place depending on the refinement of the background grid, leading to different mesh topologies as n is increased, we naturally expect some amount of noise in ρ. For

[7] The chosen multiphase elliptic interface problem has a somewhat mild coefficient jump of a factor of four across the interface. For much larger ratios, e.g., 10^3 to 10^6 and beyond, the performance degrades. In these cases, modifications to the LDG discretization can improve accuracy, conditioning, and multigrid performance significantly; see, [43].

the majority of grid sizes, we see that the convergence factor ρ is relatively constant, taking values $\rho \approx 0.15$–0.3 reflective of the challenging interface geometry; meanwhile, for the largest mesh corresponding to $n = 1024$, the slight increase in the convergence rate ρ for all p is attributed to the increased ill-conditioning of the system.

5. Concluding remarks. We have presented an *hp*-multigrid method for LDG discretizations of elliptic problems that is based on coarsening the discrete gradient and divergence operators from the flux formulation. We have shown that coarsening fine-grid operators in this way results in a method that is equivalent to pure geometric multigrid, but avoids the need to compute quantities associated with coarse meshes, such as lifting operators and quadrature rules. Whereas traditional Galerkin operator coarsening applied to the primal formulation exhibits poor multigrid performance, operator coarsening applied to the flux formulation performs well—convergence factors are nearly independent of both mesh size h and polynomial order p for the demonstrated test problems on uniform Cartesian grids, adaptively refined meshes, and implicitly defined meshes on complex geometries.

Though most of our analysis has focused on the LDG method, we believe that the essential observation applies to other forms of DG discretization of elliptic problems, particularly those in which lifting operators enter the numerical flux for $\boldsymbol{q}$. A more thorough analysis for other DG methods and more general choices of numerical fluxes would be required to determine whether the multigrid method described here extends to other methods, such as CDG or HDG. Similarly, though we have employed equal-order elements in this work, i.e., polynomials of the same degree for both u and $\boldsymbol{q}$, operator-coarsening for mixed-order elements [13] would be an interesting topic for future investigation.

We considered in this work structured meshes (Cartesian, quadtree, and octree meshes) as well as semiunstructured, nonconforming, implicitly defined meshes that result from cell merging procedures (see Figures 4.7 and 4.8). Applying the multigrid ideas presented here to problems involving more general unstructured meshes is currently under investigation. In this setting, it may be worthwhile to consider different types of relaxation methods owing to their critical role in the overall efficacy of a multigrid method. For example, additive Schwarz smoothers have been shown to be effective on nonnested polygonal meshes resulting from agglomeration procedures [5]; these smoothers could be studied in the flux coarsening context as well.

The idea of coarsening the divergence and gradient operators separately may also be useful for AMG methods, which currently treat the discrete Laplacian operator in its entirety as a black box. Indeed, black-box AMG algorithms applied to LDG discretizations appear to struggle [30]; perhaps applying AMG separately to the divergence and gradient operators in the flux formulation may yield better results.

REFERENCES

[1] P. Antonietti, B. Ayuso de Dios, S. Brenner, and L.-y. Sung, *Schwarz methods for a preconditioned WOPSIP method for elliptic problems*, Comput. Methods Appl. Math., 12 (2012), pp. 241–272, https://doi.org/10.2478/cmam-2012-0021.

[2] P. Antonietti, M. Sarti, and M. Verani, *Multigrid algorithms for hp-discontinuous Galerkin discretizations of elliptic problems*, SIAM J. Numer. Anal., 53 (2015), pp. 598–618, https://doi.org/10.1137/130947015.

[3] P. F. Antonietti, P. Houston, X. Hu, M. Sarti, and M. Verani, *Multigrid algorithms for hp-version interior penalty discontinuous Galerkin methods on polygonal and polyhedral meshes*, Calcolo, 54 (2017), pp. 1169–1198, https://doi.org/10.1007/s10092-017-0223-6.

[4] P. F. Antonietti, P. Houston, and G. Pennesi, *Fast numerical integration on polytopic meshes with applications to discontinuous Galerkin finite element methods*, J. Sci. Comput., 77 (2018), pp. 1339–1370, https://doi.org/10.1007/s10915-018-0802-y.

[5] P. F. Antonietti and G. Pennesi, *V-cycle multigrid algorithms for discontinuous Galerkin methods on non-nested polytopic meshes*, J. Sci. Comput., 78 (2019), pp. 625–652, https://doi.org/10.1007/s10915-018-0783-x.

[6] D. N. Arnold, *An interior penalty finite element method with discontinuous elements*, SIAM J. Numer. Anal., 19 (1982), pp. 742–760, https://doi.org/10.1137/0719052.

[7] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM J. Numer. Anal., 39 (2002), pp. 1749–1779, https://doi.org/10.1137/S0036142901384162.

[8] F. Bassi, A. Ghidoni, S. Rebay, and P. Tesini, *High-order accurate p-multigrid discontinuous Galerkin solution of the Euler equations*, Int. J. Numer. Methods Fluids, 60 (2008), pp. 847–865, https://doi.org/10.1002/fld.1917.

[9] F. Bassi and S. Rebay, *A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations*, J. Comput. Phys., 131 (1997), pp. 267–279, https://doi.org/10.1006/jcph.1996.5572.

[10] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, and M. Savini, *A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows*, in Proceedings of the 2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics, Technologisch Instituut, Antwerpen, Belgium, 1997, pp. 99–109.

[11] P. Bastian, M. Blatt, and R. Scheichl, *Algebraic multigrid for discontinuous Galerkin discretizations of heterogeneous elliptic problems*, Numer. Linear Algebra Appl., 19 (2012), pp. 367–388, https://doi.org/10.1002/nla.1816.

[12] S. C. Brenner and J. Zhao, *Convergence of multigrid algorithms for interior penalty methods*, Appl. Numer. Anal. Comput. Math., 2 (2005), pp. 3–18, https://doi.org/10.1002/anac.200410019.

[13] F. Brezzi, T. J. R. Hughes, L. D. Marini, and A. Masud, *Mixed discontinuous Galerkin methods for darcy flow*, J. Sci. Comput., 22 (2005), pp. 119–145, https://doi.org/10.1007/s10915-004-4150-8.

[14] F. Brezzi, G. Manzini, D. Marini, P. Pietra, and A. Russo, *Discontinuous Galerkin approximations for elliptic problems*, Numerical Methods for Partial Differential Equations, 16 (2000), pp. 365–378, https://doi.org/10.1002/1098-2426(200007)16:4<365::AID-NUM2>3.0.CO;2-Y.

[15] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial, Second Edition*, SIAM, Philadelphia, 2000, https://doi.org/10.1137/1.9780898719505.

[16] P. Castillo, B. Cockburn, I. Perugia, and D. Schötzau, *An a priori error analysis of the local discontinuous Galerkin method for elliptic problems*, SIAM J. Numer. Anal., 38 (2000), pp. 1676–1706, https://doi.org/10.1137/S0036142900371003.

[17] B. Cockburn and B. Dong, *An analysis of the minimal dissipation local discontinuous Galerkin method for convection–diffusion problems*, J. Sci. Comput., 32 (2007), pp. 233–262, https://doi.org/10.1007/s10915-007-9130-3.

[18] 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, https://doi.org/10.1137/070706616.

[19] B. Cockburn, G. Kanschat, I. Perugia, and D. Schötzau, *Superconvergence of the local discontinuous Galerkin method for elliptic problems on Cartesian grids*, SIAM J. Numer. Anal., 39 (2001), pp. 264–285, https://doi.org/10.1137/S0036142900371544.

[20] B. Cockburn and C.-W. Shu, *The local discontinuous Galerkin method for time-dependent convection-diffusion systems*, SIAM J. Numer. Anal., 35 (1998), pp. 2440–2463, https://doi.org/10.1137/S0036142997316712.

[21] J. Douglas and T. Dupont, *Interior penalty procedures for elliptic and parabolic Galerkin methods*, in Computing Methods in Applied Sciences, Lecture Notes in Phys. 58, R. Glowinski and J. L. Lions, eds., Springer, Berlin, 1976, pp. 207–216, https://doi.org/10.1007/BFb0120591.

[22] M. S. Fabien, M. G. Knepley, R. T. Mills, and B. M. Riviere, *Manycore parallel computing for a hybridizable discontinuous Galerkin nested multigrid method*, SIAM J. Sci. Comput., 41 (2019), pp. C73–C96, https://doi.org/10.1137/17M1128903.

[23] K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal, *p-multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations*, J. Comput. Phys., 207 (2005), pp. 92–113, https://doi.org/10.1016/j.jcp.2005.01.005.

[24] J. Gopalakrishnan and G. Kanschat, *A multilevel discontinuous Galerkin method*, Numer. Math., 95 (2003), pp. 527–550, https://doi.org/10.1007/s002110200392.

[25] B. T. Helenbrook, D. Mavriplis, and H. L. Atkins, *Analysis of p-multigrid for Continuous and Discontinuous Finite Element Discretizations*, in Proceedings of the 16th AIAA Computational Fluid Dynamics Conference, 2003, https://doi.org/10.2514/6.2003-3989.

[26] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Texts Appl. Math. 54, Springer, New York, 2008, https://doi.org/10.1007/978-0-387-72067-8.

[27] G. Kanschat, *Preconditioning methods for local discontinuous Galerkin discretizations*, SIAM J. Sci. Comput., 25 (2003), pp. 815–831, https://doi.org/10.1137/S1064827502410657.

[28] G. Kanschat, *Multilevel methods for discontinuous Galerkin FEM on locally refined meshes*, Comput. Struct., 82 (2004), pp. 2437–2445, https://doi.org/10.1016/j.compstruc.2004.04.015.

[29] H. Luo, J. D. Baum, and R. Löhner, *A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids*, J. Comput. Phys., 211 (2006), pp. 767–783, https://doi.org/10.1016/j.jcp.2005.06.019.

[30] L. N. Olson and J. B. Schroder, *Smoothed aggregation multigrid solvers for high-order discontinuous Galerkin methods for elliptic problems*, J. Comput. Phys., 230 (2011), pp. 6959–6976, https://doi.org/10.1016/j.jcp.2011.05.009.

[31] J. Peraire and P.-O. Persson, *The compact discontinuous Galerkin (CDG) method for elliptic problems*, SIAM J. Sci. Comput., 30 (2008), pp. 1806–1824, https://doi.org/10.1137/070685518.

[32] P.-O. Persson, *A sparse and high-order accurate line-based discontinuous Galerkin method for unstructured meshes*, J. Comput. Phys., 233 (2013), pp. 414–429, https://doi.org/10.1016/j.jcp.2012.09.008.

[33] I. Perugia and D. Schötzau, *An hp-analysis of the local discontinuous Galerkin method for diffusion problems*, J. Sci. Comput., 17 (2002), pp. 561–571, https://doi.org/10.1023/A:1015118613130.

[34] F. Prill, M. Lukáčová-Medviďová, and R. Hartmann, *Smoothed aggregation multigrid for the discontinuous Galerkin method*, SIAM J. Sci. Comput., 31 (2009), pp. 3503–3528, https://doi.org/10.1137/080728457.

[35] R. I. Saye, *High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles*, SIAM J. Sci. Comput., 37 (2015), pp. A993–A1019, https://doi.org/10.1137/140966290.

[36] R. I. Saye, *Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part* I, J. Comput. Phys., 344 (2017), pp. 647–682, https://doi.org/10.1016/j.jcp.2017.04.076.

[37] R. I. Saye, *Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part* II, J. Comput. Phys., 344 (2017), pp. 683–723, https://doi.org/10.1016/j.jcp.2017.05.003.

[38] C. Siefert, R. Tuminaro, A. Gerstenberger, G. Scovazzi, and S. S. Collis, *Algebraic multigrid techniques for discontinuous Galerkin methods with varying polynomial order*, Comput. Geosci., 18 (2014), pp. 597–612, https://doi.org/10.1007/s10596-014-9419-x.

[39] J. Stiller, *Robust multigrid for high-order discontinuous Galerkin methods: A fast Poisson solver suitable for high-aspect ratio Cartesian grids*, J. Comput. Phys., 327 (2016), pp. 317–336, https://doi.org/10.1016/j.jcp.2016.09.041.

[40] M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, *An adaptive level set approach for incompressible two-phase flows*, J. Comput. Phys., 148 (1999), pp. 81–124, https://doi.org/10.1006/jcph.1998.6106.

[41] O. Tatebe, *The multigrid preconditioned conjugate gradient method*, in Sixth Copper Mountain Conference on Multigrid Methods, NASA Langley Research Center, Hampton, VA, 1993, pp. 621–634.

[42] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613, https://doi.org/10.1137/1034116.

[43] R. Saye, *Efficient Multigrid Solution of Elliptic Interface Problems using Viscosity-Upwinded Local Discontinuous Galerkin Methods*, Commun. Appl. Math. Comput. Sci., 2019, in press.

[44] J. Yan and C.-W. Shu, *Local discontinuous Galerkin methods for partial differential equations with higher order derivatives*, J. Sci. Comput., 17 (2002), pp. 27–47, https://doi.org/10.1023/A:1015132126817.