LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-824455

On ParELAG's Parallel Element-based Algebraic Multigrid and its MFEM Miniapps for H(curl) and H(div) Problems: a report including lowest and next to the lowest order numerical results

D. Z. Kalchev, P. S. Vassilevski, U. Villa

July 13, 2021

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

ON PARELAG'S PARALLEL ELEMENT-BASED ALGEBRAIC MULTIGRID AND ITS MFEM MINIAPPS FOR $H(\underline{\text{curl}})$ AND $H(\text{div})$ PROBLEMS:

A REPORT INCLUDING LOWEST AND NEXT TO THE LOWEST ORDER NUMERICAL RESULTS

DELYAN Z. KALCHEV, PANAYOT S. VASSILEVSKI, AND UMBERTO VILLA

Abstract. This paper presents the utilization of element-based algebraic multigrid (AMGe) hierarchies, implemented in the ParELAG (Parallel Element Agglomeration Algebraic Multigrid Upscaling and Solvers) library, to produce multilevel preconditioners and solvers for $H(\underline{\text{curl}})$ and $H(\text{div})$ formulations. This involves the construction of hierarchies of compatible nested spaces, forming an exact de Rham sequence on each level. This allows the application of hybrid smoothers on all levels and AMS (Auxiliary-space Maxwell Solver) or ADS (Auxiliary-space Divergence Solver) on the coarsest levels, obtaining complete multigrid cycles. Numerical results are presented, showing the parallel performance of the proposed methods. As a part of the exposition, this paper demonstrates some of the capabilities of ParELAG and outlines some of the components and procedures within the library.

Key words. algebraic multigrid (AMG), AMGe, $H(\underline{\text{curl}})$ solvers, $H(\text{div})$ solvers, ADS, AMS, de Rham sequence, hybrid smoothers, finite element methods, ParELAG, MFEM

Mathematics subject classification. 65F08, 65F10, 65N22, 65N30, 65N55

1. Introduction

In a number of problems in physics and engineering, that are addressed by advanced scientific computing methods, arise models and formulations involving the curl (rotation) and divergence operators. As a part of preconditioning or solving systems of partial differential equations (PDEs) and multiphysics simulations, finite element forms and blocks posed on the spaces $H(\underline{\text{curl}})$ and $H(\text{div})$ emerge. For example, this includes models of electromagnetism using Maxwell equations (possibly as a part of larger multiphysics codes) [48, 56, 15], mixed finite element methods for second-order elliptic equations [17] and coupled systems [60, 8], first-order system least-squares (FOSLS) finite element methods [20, 54, 6, 7], certain formulations of the Stokes and Navier-Stokes equations [21, 22, 47], and radiation transport simulations [19].

The solution of the resulting linear systems involves inversion or preconditioning of $H(\underline{\text{curl}})$ and $H(\text{div})$ forms. One major source of difficulty is the large null spaces of the curl and divergence. A variety of approaches have been developed, including multigrid methods, both geometric and algebraic multigrid (AMG) [29, 55, 14, 34, 28, 10, 27, 13, 61, 9, 62, 58], methods reducing the problem to (geometric) multigrid [31, 39], a recent approach employing hybridization and static condensation [24], and methods in the field of domain decomposition [49, 63, 64, 52, 33]. Hiptmair and Xu [32] proposed auxiliary space preconditioners employing stable regular decompositions;

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-TR-824455).

The work of the second author was partially supported by NSF under grant DMS-1619640.

see also [16]. Based on those ideas quite successful parallel $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$ solvers were developed as a part of HYPRE [1]: AMS [40] and ADS [41]. Furthermore, recent developments of generic auxiliary space preconditioners [38, 37], utilizing nonconforming reformulations and static condensation, can potentially be employed to implement efficient preconditioners for $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$ problems.

This paper describes and demonstrates the utilization of an AMGe (element-based AMG; see [35, 57, 58]) approach for preconditioning conforming discrete $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$ formulations. While AMGe methods were originally developed in the context of symmetric positive definite (SPD) systems coming from H^1-conforming formulations [23, 45], they demonstrate the capacity for a broader applicability. An important role in the construction of the AMGe multilevel methods is played by the *de Rham sequence* (i.e., in three dimensions, $H^1 \to H(\underline{\mathrm{curl}}) \to H(\mathrm{div}) \to L^2$) of Sobolev spaces. It provides a quite elegant tool in the theory of finite elements for a variety of problems; see [30, 11]. Namely, a discrete version of the sequence of conforming finite element spaces, maintaining the *exactness* and *commutativity* properties, delivers numerical stability (inf-sup compatibility) for a variety of mixed finite element methods. These ideas are used and investigated in [46, 44, 53] for the construction of multilevel element-based algebraic hierarchies of de Rham sequences of spaces. The last constitutes the foundation for the current work expounded in this paper.

A fundamental idea in AMGe, as presented in this work and implemented in ParELAG [5], is the element-based construction of coarse levels that structurally resemble (fine) geometric levels composed of standard finite elements. This involves the identification of coarse meshes, contrived of coarse elements, with properly established *coarse mesh topologies* in the form of relations between coarse elements and coarse lower-dimensional mesh entities (facets, edges, and vertices), similarly to geometric levels. Consequently, utilizing the coarse topology, each coarse space is built via independent local coarse-element-by-coarse-element computations, whose combined effect is a conforming global coarse space. The independence of the local work makes the construction of AMGe hierarchies naturally attuned for parallel computing.

Starting with a given fine mesh $\mathcal{T}^h$, a basic idea in AMGe, as outlined in the previous paragraph, is to build a coarse mesh $\mathcal{T}^H$ of *agglomerated elements* — non-overlapping unions of fine-level elements, which act as coarse elements. Using an algorithm for identifying *minimal intersection sets* [58], lower-dimensional coarse mesh entities, like coarse facets, edges, and vertices, in $\mathcal{T}^H$ are determined together with their relationships, forming the topology of $\mathcal{T}^H$ in virtue of $\mathcal{T}^h$. The mesh topology is needed for the construction of the de Rham sequences as it reflects a natural structure within the sequence of spaces and the transitional differential operators (*exterior derivatives*). In accordance with the ideas of multigrid, the coarse de Rham sequence (of coarse spaces) is formed in terms of the fine one as a sequence of subspaces, i.e., coarse basis functions are linear combinations of fine basis functions. Consequently, the coarsening procedure is algebraic in nature, involving the construction of coarse basis functions as algebraic vectors and organizing them in *prolongation matrices*. The building of coarse bases is an intricate procedure, entailing the obtainment of *target traces*, or shortly *targets*, (e.g., in $H(\mathrm{div})$, these are normal flux traces on the coarse facets) and a two stage *extension* process, involving the solution of small local, on coarse entities, mixed finite element problems to produce the final coarse basis functions. First, the traces are gradually extended to higher-dimensional entities (e.g., in $H(\mathrm{div})$, the normal flux traces are extended from

coarse facets to whole coarse elements) and then an "extension" across spaces, in reverse direction of the de Rham sequence, is performed to maintain compatibility (exactness) between the spaces and operators in the sequence. Here, we concentrate on polynomial targets, while spectral targets (i.e., coming from solving local generalized eigenvalue problems) can potentially also be used [36]. Note that the obtained coarse spaces maintain the general finite element structure of the fine spaces. Therefore, as typical in AMGe, the coarsening can be performed recursively, producing a hierarchy of nested and compatible, on each level, spaces.

The above shortly described methodology is implemented in ParELAG. This is a parallel library that builds hierarchies of stable sequences of discrete spaces with approximation properties, to be utilized typically as discretization tools for numerical upscaling [59] of mixed finite element formulations. It also provides a set of respective preconditioners and solvers that can be used for solving the resulting problems or building further intricate solvers. ParELAG has been successfully applied, e.g., in upscaling for reservoir modelling [43] and multilevel Monte Carlo simulations [51, 50, 25, 26].

This paper discusses the construction of multilevel solvers for $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$ problems, using the hierarchies of spaces from ParELAG. The availability of entire de Rham sequences, together with all necessary transfer operators, on all levels allows the utilization of *hybrid (Hiptmair) smoothers* [29, 58] on all levels, as well as AMS and ADS on the coarsest levels, producing complete multigrid cycles. An outline of the overall methodology is presented and the parallel performance of the proposed solvers is shown in numerical examples. Part of the goal of this work is to exhibit ParELAG and some of its capabilities. Therefore, as a demonstration, mini applications, that invoke ParELAG for solving $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$ problems, are produced within MFEM [3] — a finite element library, that is a subject of a recent increase in popularity and becoming a tool of choice. As a part of the exposition in the paper, describing the methods, a basic overview of some constructs within ParELAG is provided, as utilized in the mentioned mini applications. To fix the presentation, the ideas are conveyed for the three-dimensional case, while one can easily see how they would be applied in a two-dimensional setting.

The outline of the remainder of the paper is as follows. The $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$ problems of interest are presented in Section 2. Section 3 is devoted to an overview of the methodology, involving the de Rham sequence, the above mentioned coarsening and extension procedure, and the transfer operators. Those operators are useful for implementing the hybrid smoothers on all levels and the AMS and ADS coarse solvers, as described in Section 4. Numerical results, demonstrating the parallel performance of the proposed methods, are in Section 5. The conclusions and a discussion on possible future directions are left for the last Section 6.

2. Problems formulations

This section provides a basic presentation of spaces, some notation, and formulations. Let $\Omega \subset \mathbb{R}^3$ be a bounded contractible[1] domain with a Lipschitz-continuous boundary $\Gamma = \partial\Omega$. Define the spaces of square-integrable vector fields on Ω with, respectively, square-integrable curl, $H(\underline{\mathrm{curl}})$, and square-integrable divergence, $H(\mathrm{div})$, as

$$H(\underline{\mathrm{curl}};\ \Omega) = \left\{ \underline{v} \in [L^2(\Omega)]^3;\ \underline{\mathrm{curl}}\, \underline{v} \in [L^2(\Omega)]^3 \right\},$$

[1]Recall that this intuitively means that Ω has no holes and can be continuously contracted into a point. That is, Ω is homotopy equivalent to a ball and to a single point.

$$H(\mathrm{div};\ \Omega) = \big\{\, \underline{v} \in [L^2(\Omega)]^3;\ \mathrm{div}\, \underline{v} \in L^2(\Omega) \,\big\}.$$

These are Hilbert spaces endowed with the respective norms $\|\underline{v}\|_{\underline{\mathrm{curl}}} = (\|\underline{v}\|_0^2 + \|\underline{\mathrm{curl}}\, \underline{v}\|_0^2)^{1/2}$ and $\|\underline{v}\|_{\mathrm{div}} = (\|\underline{v}\|_0^2 + \|\mathrm{div}\, \underline{v}\|_0^2)^{1/2}$, where $\|\cdot\|_0$ denotes the norms in both $L^2(\Omega)$ and $[L^2(\Omega)]^3$. Introduce also the space $H^1(\Omega) = \{\, v \in L^2(\Omega);\ \underline{\mathrm{grad}}\, v \in [L^2(\Omega)]^3 \,\}$, of square-integrable functions with square integrable first derivatives, which is a Hilbert space with a norm $\|v\|_{\underline{\mathrm{grad}}} = (\|v\|_0^2 + \|\underline{\mathrm{grad}}\, v\|_0^2)^{1/2}$. For consistency, it can also be denoted by $H(\underline{\mathrm{grad}};\ \Omega) = H^1(\Omega)$. Also, $L^2(\Omega)$ can be conformed to this notation convention by alternatively denoting it as $H(0;\ \Omega) = L^2(\Omega)$, where in such a context 0 represents a generic zero operator. For convenience, "Ω" can be skipped in the notation, which should not lead to any confusion. This allows the deployment of a generic notation, using $D \in \{\, \underline{\mathrm{grad}}, \underline{\mathrm{curl}}, \mathrm{div}, 0 \,\}$ and considering the respective space $H(D)$ with a norm $\|\cdot\|_D$.

Consider the symmetric bilinear forms of interest

$$\begin{aligned} a_{\underline{\mathrm{curl}}}(\underline{u}, \underline{v}) &= (\alpha\ \underline{\mathrm{curl}}\, \underline{u}, \underline{\mathrm{curl}}\, \underline{v})_0 + (\beta\, \underline{u}, \underline{v})_0 \quad \text{for } \ \underline{u}, \underline{v} \in H(\underline{\mathrm{curl}};\ \Omega), \\ a_{\mathrm{div}}(\underline{u}, \underline{v}) &= (\alpha\ \mathrm{div}\, \underline{u}, \mathrm{div}\, \underline{v})_0 \ + (\beta\, \underline{u}, \underline{v})_0 \quad \text{for } \ \underline{u}, \underline{v} \in H(\mathrm{div};\ \Omega), \end{aligned} \tag{2.1}$$

where $\alpha, \beta \in L^\infty(\Omega)$, $\alpha > 0$, $\beta > 0$, and $(\cdot,\cdot)_0$ denotes the inner products in both $L^2(\Omega)$ and $[L^2(\Omega)]^3$. The bilinear forms are positive definite and posses coefficient-dependent continuity in term of $\|\cdot\|_{\underline{\mathrm{curl}}}$ and $\|\cdot\|_{\mathrm{div}}$, respectively. If the coefficients are bounded away from zero, then the bilinear forms satisfy respective coefficient-dependent coercivity.

Remark 2.1. One can actually consider $\beta \geq 0$, in which case the bilinear forms in (2.1) are generally semi-definite since they are singular in regions where $\beta = 0$. For simplicity in the examples here, we use $\beta > 0$. More generally, β can be an essentially bounded symmetric positive (semi-)definite tensor. The bilinear forms are positive definite when β is positive definite, generally semi-definite when β is semi-definite, and coercivity depends on β being uniformly (on Ω) positive definite.

Problems involving (2.1), arising in practice, are posed on corresponding conforming discrete finite element spaces. This results in discrete versions of the bilinear forms, represented by respective symmetric positive (semi-)definite matrices. The goal in this paper is to present preconditioners for linear systems with such matrices. The corresponding conforming finite element spaces, defined on a given fine mesh $\mathcal{T}^h$, are denoted by $\mathcal{V}^h(\underline{\mathrm{grad}}) \subset H(\underline{\mathrm{grad}})$, $\mathcal{V}^h(\underline{\mathrm{curl}}) \subset H(\underline{\mathrm{curl}})$, $\mathcal{V}^h(\mathrm{div}) \subset H(\mathrm{div})$, and $\mathcal{V}^h(0) \subset H(0)$. They are spaces of, respectively, continuous piecewise polynomial Lagrangian (nodal), Nédélec, Raviart–Thomas, and discontinuous piecewise polynomial finite elements [17]. In the case of lowest order finite elements, the degrees of freedom (dofs) in the spaces are associated with mesh entities of increasing dimensionality, one dof per entity. Namely, these are, respectively, point values at vertices, tangential flow along edges, normal flux across facets, constant values in elements (sometimes referred to as cells).

3. Overview of the multilevel de Rham sequence

Here, the basics of the de Rham sequence of interest, its discrete version, the coarsening methodology, and the involved operators and procedures are presented and discussed. While the topic is the preconditioning of $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$ formulations, the methodology here, as it becomes clear in Section 4, needs the consideration and utilization of the entire de Rham complex.

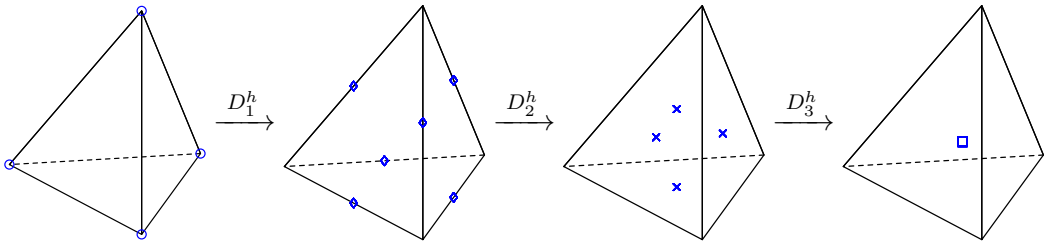


FIGURE 3.1. An illustration of the mapping between dofs in a tetrahedral element for the lowest order case.

3.1. **De Rham sequences of spaces.** Consider the de Rham complex of Sobolev spaces on Ω together with a respective *subcomplex* of conforming finite element spaces

$$\begin{array}{ccccccccccc} \mathbb{R} & \xrightarrow{D_0=\mathfrak{I}} & H(D_1) & \xrightarrow{D_1=\underline{\mathrm{grad}}} & H(D_2) & \xrightarrow{D_2=\underline{\mathrm{curl}}} & H(D_3) & \xrightarrow{D_3=\mathrm{div}} & H(D_4) & \xrightarrow{D_4=0} & \{0\} \\ & & \downarrow{\scriptstyle \Pi_1^h} & & \downarrow{\scriptstyle \Pi_2^h} & & \downarrow{\scriptstyle \Pi_3^h} & & \downarrow{\scriptstyle \Pi_4^h} & & \\ \mathbb{R} & \xrightarrow{D_0^h=\mathfrak{I}} & \mathcal{V}^h(D_1) & \xrightarrow{D_1^h} & \mathcal{V}^h(D_2) & \xrightarrow{D_2^h} & \mathcal{V}^h(D_3) & \xrightarrow{D_3^h} & \mathcal{V}^h(D_4) & \xrightarrow{D_4^h=0} & \{0\} \end{array} , \tag{3.1}$$

where "$\mathbb{R}$" represents the constants, $\mathfrak{I}$ is simply an injection that maps a real number to a corresponding constant function on Ω, $\Pi_i^h\colon H(D_i) \to \mathcal{V}^h(D_i)$ for $i = 1, \dots, 4$ are respective appropriate *(cochain)* projection operators, $D_i\colon H(D_i) \to H(D_{i+1})$ for $i = 1, \dots, 3$ are the respective differential operators mapping between the Sobolev spaces, and $D_i^h\colon \mathcal{V}^h(D_i) \to \mathcal{V}^h(D_{i+1})$ are their corresponding discrete versions with matching semantics but formulated on the finite element spaces. Clearly, in the finite-dimensional setting, the functions in $\mathcal{V}^h(D_i)$ for $i = 1, \dots, 4$ can be identified with algebraic vectors defined on the respective dofs. Hence, D_i^h for $i = 1, \dots, 3$ can be viewed as matrices in $\mathbb{R}^{d_{i+1}^h \times d_i^h}$, where $d_i^h = \dim(\mathcal{V}^h(D_i))$, expressed in terms of the bases in $\mathcal{V}^h(D_i)$. They can be assembled via an *overwriting* finite element assembly[2] procedure from local, on elements, versions of the operators and their matrices. Particularly, in the lowest order case, D_i^h from left to right map mesh entities from lower to higher dimensionality, i.e, vertices $\to$ edges, edges $\to$ facets, and facets $\to$ elements, respectively; see Fig. 3.1. The matrices for D_i^h are provided by MFEM [3]. It is assumed that Π_i^h for $i = 1, \dots, 4$ are bounded operators, i.e., $\|\Pi_i^h\|_{D_i} < \infty$, where $\|\cdot\|_{D_i}$ also denote the corresponding induced operator norms. This holds for the considered finite element spaces and implies the quasi-optimality property $\|u - \Pi_i^h u\|_{D_i} \le \|I - \Pi_i^h\|_{D_i} \inf_{v^h \in \mathcal{V}^h(D_i)} \|u - v^h\|_{D_i}$ for all $u \in H(D_i)$.

Observe that it generally holds that $D_{i+1}D_i = 0$ (i.e., $\mathrm{Range}(D_i) \subset \mathrm{Ker}(D_{i+1})$) for $i = 0, \dots, 3$. The sequence is called *exact* if $\mathrm{Range}(D_i) = \mathrm{Ker}(D_{i+1})$ for $i = 0, \dots, 3$, which depends on the topological characteristics of Ω. The connectivity of Ω is sufficient to demonstrate this property for $i = 0$ and 3, whereas it holds for $i = 1$ using that Ω is simply-connected. The contractibility of Ω provides the property for $i = 2$ as a consequence of Poincaré's lemma; see, e.g., [30]. The discrete subcomplex of spaces $\mathcal{V}^h(D_i)$ also mirrors the exactness property. This depends on the corresponding topological characteristics of the mesh cells. Therefore, we also require that the cells and their facets and edges be contractible.

The null space of the gradient, D_1, can be eliminated by considering the quotient spaces $H(D_1)/\mathbb{R}$ and $\mathcal{V}^h(D_1)/\mathbb{R}$ of functions with a zero mean and replacing "$\mathbb{R}$" in

[2] *Overwriting* means that during the assembly the entries in the global matrix are overwritten by the values of the entries in the local matrices rather than accumulating (adding) them.

(3.1) with "$\{\,0\,\}$", turning the de Rham sequence into $\{\,0\,\} \to H(D_1)/\mathbb{R} \to \cdots \to H(D_4) \to \{\,0\,\}$. This is also the case when boundary conditions[3] are imposed on the functions in $H(D_1)$ and $\mathcal{V}^h(D_1)$ on a portion of the boundary $\Gamma_0 \subset \Gamma$. Note that this affects the de Rham sequence. Recall that the natural notion of a *trace*, denoted by a trace operator γ_1, in $H(D_1)$ is an extension of the notion of a function value (a restriction of a function) on a surface, the trace γ_2 in $H(D_2)$ extends the tangential flow, $\underline{v} \times \underline{n}$, along a surface, and the trace γ_3 in $H(D_3)$ extends the normal flux, $\underline{v} \cdot \underline{n}$, across a surface. Here, $\underline{n}$ denotes an appropriate outward unit normal vector to the surface (portion of the boundary). Thus, using a convenient intuitive notation, working with a space of functions $v \in H(D_1)$ such that $\gamma_1 v|_{\Gamma_0} = 0$ requires considering, in the de Rham sequence, spaces of vector fields $\underline{v} \in H(D_2)$ such that $\gamma_2 \underline{v}|_{\Gamma_0} = 0$ and $\underline{v} \in H(D_3)$ such that $\gamma_3 \underline{v}|_{\Gamma_0} = 0$. This is mirrored by the discrete subcomplex. However, this is mostly a formality. Note that, for the utilized finite elements, such boundary conditions are mapped and imposed directly on respective dofs associated with the boundary, allowing to be viewed as respective *essential* boundary conditions, as needed, in the context of Galerkin-type formulations, i.e., boundary conditions that are explicitly imposed on the spaces, via the elimination of the corresponding boundary dofs in the respective matrices, rather than as a *natural* part of a variational formulation.

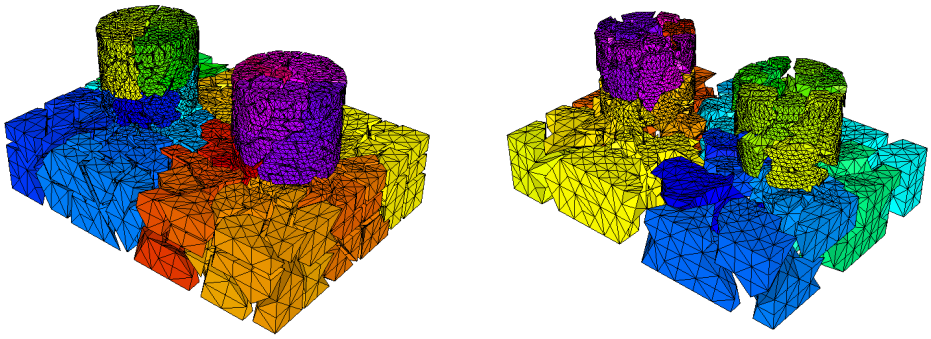
Remark 3.1. A special case is when the boundary conditions are posed on the entire boundary Γ, obtaining the spaces $H_0(D_i)$ for $i = 1, \ldots, 3$. Since the exactness is related to the divergence theorem $\int_\Omega \operatorname{div} \underline{v} \, \mathrm{d}\underline{x} = \int_\Gamma \gamma_3 \underline{v} \, \mathrm{d}\sigma = 0$ for $\underline{v} \in H_0(D_3)$, one needs to either consider $H_0(D_4) = H(D_4)/\mathbb{R}$ in (3.1), or consider the entire $H(D_4) = L^2(\Omega)$ but formally replace the zero operator, $D_4 = 0$, with the integral on Ω, whose null space is precisely the functions with a zero mean, and further replace "$\{\,0\,\}$" in (3.1) with "$\mathbb{R}$", turning the de Rham sequence into $\{\,0\,\} \to H_0(D_1) \to H_0(D_2) \to H_0(D_3) \to H(D_4) \to \mathbb{R}$.

Importantly, for the used finite element spaces, the following *commutativity* property holds:

$$D_i^h \circ \Pi_i^h = \Pi_{i+1}^h \circ D_i \quad \text{for } i = 1, \ldots, 3. \tag{3.2}$$

That is, the *diagram* in (3.1) is *commutative*. The exactness of the continuous de Rham complex provides, e.g., stable decompositions (like the Helmholtz decomposition [48] and the so-called regular ones in [32]), while the commutativity of (3.1) and the exactness of the discrete subcomplex contribute to the inheritance of some important properties in the discrete setting, like the discrete stable decompositions in [32] and the provision of the (inf-sup) stability of certain mixed finite element methods; see [17, 30, 11]. Such stability, together with the approximation properties of the discrete spaces, are important for the convergence of those mixed finite element methods. ParELAG, which is intended both as a discretization tool and for the construction of multigrid solvers, builds de Rham sequences of coarse spaces satisfying the exactness and commutativity properties with the goal of further maintaining important characteristics (like stability [46, 44, 53]), that are innate to the fine sequence, on coarse levels. While no discretizations of mixed finite element formulations are explicitly considered in this work, these features of ParELAG are potentially beneficial since both the solvers in [32, 40, 41] and the utilized hybrid smoothers [29][58, Appendix F] are formulated with a fine-like finite element construct, i.e., geometric-like spaces, in mind. The element-based multilevel approach

[3]Without loss of generality, all boundary conditions are considered homogeneous (i.e., zero).



(A) Agglomerates of fine elements (B) Agglomerates of agglomerates

FIGURE 3.2. Examples of agglomerates. (The gaps between agglomerates are for illustrative purposes.)

implemented in ParELAG provides such a geometric-like finite element structure on all levels.

3.2. **Coarse de Rham sequences.** The fundamentals and notions associated with the coarsening of the de Rham sequence are now described. This also serves as a preparation for Section 3.3.

While ParELAG can concentrate only on a portion, $\{ H(D_i) \}_{i=s}^{4}$ for any $s = 1, \dots, 4$, of the complex, the entire sequence is discussed here as needed. Consider the de Rham complex of fine-scale spaces together with a respective subcomplex of coarse subspaces

$$
\begin{array}{ccccccccccc}
\mathbb{R} & \xrightarrow{D_0^h=\mathfrak{I}} & \mathcal{V}^h(D_1) & \xrightarrow{D_1^h} & \mathcal{V}^h(D_2) & \xrightarrow{D_2^h} & \mathcal{V}^h(D_3) & \xrightarrow{D_3^h} & \mathcal{V}^h(D_4) & \xrightarrow{D_4^h=0} & \{0\} \\
 & & \downarrow{\scriptstyle \Pi_1^H} & & \downarrow{\scriptstyle \Pi_2^H} & & \downarrow{\scriptstyle \Pi_3^H} & & \downarrow{\scriptstyle \Pi_4^H} & & \\
\mathbb{R} & \xrightarrow{D_0^H=\mathfrak{I}} & \mathcal{V}^H(D_1) & \xrightarrow{D_1^H} & \mathcal{V}^H(D_2) & \xrightarrow{D_2^H} & \mathcal{V}^H(D_3) & \xrightarrow{D_3^H} & \mathcal{V}^H(D_4) & \xrightarrow{D_4^H=0} & \{0\}
\end{array}
\tag{3.3}
$$

3.2.1. *The coarse mesh.* The first step is the generation of a coarse mesh $\mathcal{T}^H$, from the given fine one $\mathcal{T}^h$, including all mesh entities: coarse elements, facets, edges, and vertices. The foundation of this is the construction of coarse elements as *agglomerates* (or *agglomerated elements*) T, which provide a non-overlapping partition of the fine elements; see Fig. 3.2a. One customary way to achieve that is via partitioning (e.g., using METIS [2]) of the *dual graph* of $\mathcal{T}^h$ — a graph whose nodes are the elements in $\mathcal{T}^h$ and any two nodes are connected in the graph when the respective mesh elements share a facet. It is not difficult to generate the agglomerates as contiguous partitions in terms of the dual graph, e.g., using METIS or simply identifying the connected components of the partitioning after it is generated. Moreover, ParELAG provides additional tools that can help, via weighting the dual graph and further splitting of agglomerates, improve the topological properties of the coarse elements, which are relevant if $H(\underline{\text{curl}})$ is utilized. ParELAG contains a set of *partitioner* classes, which generate an element partitioning on the current level that composes the agglomerated elements. For example, the class `MFEMRefinedMeshPartitioner` constructs agglomerates in the form of geometric coarse elements by reverting previous refinements performed by MFEM, while the class `MetisGraphPartitioner` invokes METIS internally.

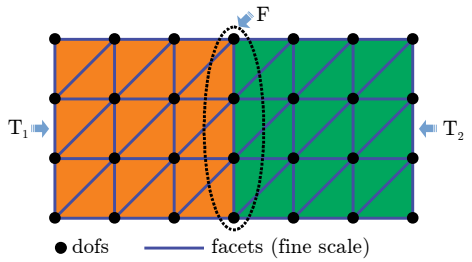


FIGURE 3.3. A two-dimensional illustration of the designation of a coarse facet F as a set of fine-scale facets, serving as an interface between agglomerates T_1 and T_2.

Using the partitioning of $\mathcal{T}^h$ and viewing each agglomerate $T \in \mathcal{T}^H$ as a collection of fine facets, an intersection procedure (see [58, section 1.9]) over these collections provides the coarse facets as sets of fine facets (see Fig. 3.3), that can be consistently interpreted as interface surfaces between coarse elements. Further viewing the obtained coarse facets as collections of fine edges, their intersection identifies coarse edges as sets of fine edges. Finally, the intersection of coarse edges in terms of fine vertices identifies the coarse vertices. ParELAG collects all these coarse entities together with relationships between them in the form of so called *(agglomerated) topology* of $\mathcal{T}^H$, represented by an object of the ParELAG class `AgglomeratedTopology`. Note that such a topology object in itself also represents a complex related to (3.3). It further contains relations between agglomerated entities and their comprising fine ones. In more detail, the `AgglomeratedTopology` object on the finest level is obtained from the given mesh $\mathcal{T}^h$ (i.e., using MFEM). Having a topology on the current level, a new coarser agglomerated topology is generated by invoking the `CoarsenLocalPartitioning()` member function of `AgglomeratedTopology`, using the agglomerated elements produced on the current level by a partitioner class.

Coarse facets and edges additionally carry information about the orientation of their constituting fine entities. Such a set of fine-scale orientations for a coarse entity represents the orientation of that coarse entity. More precisely, these are $+1$ and -1 data entries in the agglomerated topology relating each coarse entity to its comprising fine-scale ones, respectively representing the preservation or the reversion of the original orientation of the fine entity within the coarse-scale one, so that each agglomerated entity has a consistent orientation. For example, a coarse facet F has an associated vector of $+1$ and -1, denoted by $\boldsymbol{\varphi}^F$, that based on the orientation of the constituting fine facets devises a consistent orientation for F, so that the normal vector to F points everywhere from one of its adjacent agglomerates to the other, e.g., from T_1 to T_2 in Fig. 3.3.

Furthermore, coarse facets associated with the domain boundary, or portions of it, are identified, thus also determining, via relationship, respective coarse edges and vertices on the boundary. This allows considering boundary dofs and boundary conditions on coarse levels. Also, the coarse elements can be optionally made to conform to material (coefficient) interfaces by splitting agglomerates that cross such interfaces.

3.2.2. *The element-based construction of the coarse bases and the prolongation matrices.* The coarse spaces $\mathcal{V}^H(D_i)$ for $i = 1, \ldots, 4$ are obtained via the construction of coarse bases as sets of algebraic vectors in terms of the respective dofs in $\mathcal{V}^h(D_i)$, i.e.,

coarse basis functions are linear combinations of fine basis functions. These algebraic vectors constitute the columns of corresponding *prolongation* matrices $P_i \in \mathbb{R}^{d_i^h \times d_i^H}$, $P_i \colon \mathcal{V}^H(D_i) \to \mathcal{V}^h(D_i)$, with full column ranks, where $d_i^H = \dim(\mathcal{V}^H(D_i))$. Similarly to the fine level, the coarse basis functions are supported locally and built by local agglomerate-by-agglomerate procedures (more details in Section 3.3). The $\mathcal{V}^H(D_i)$-dofs are identified with the columns of P_i, i.e., with the respective coarse basis functions. Moreover, the $\mathcal{V}^H(D_i)$-dofs associated with an agglomerate $T \in \mathcal{T}^H$ are the ones whose basis functions have supports intersecting T, and the restrictions of those basis functions (i.e., the coarse *shape functions*) on T are precisely the restrictions of the respective algebraic vectors on the $\mathcal{V}^h(D_i)$-dofs of T, which are the $\mathcal{V}^h(D_i)$-dofs associated with the fine elements $\tau \in \mathcal{T}^h$ such that $\tau \subset T$. Therefore, local-on-T prolongation matrices $P_{T,i}$ can be defined and they are submatrices of P_i on the respective dofs in $\mathcal{V}^h(D_i)$ and $\mathcal{V}^H(D_i)$ on T.

Next, we consider a generic bilinear form $a_{ij}(\cdot,\cdot)$ defined on $H(D_i) \times H(D_j)$ for some $i,j = 1,\dots,4$. On the conforming discrete subspaces $\mathcal{V}^h(D_i)$ and $\mathcal{V}^h(D_j)$, using their finite element bases, the bilinear form is represented by a (global) matrix $A_{ij}^h \in \mathbb{R}^{d_j^h \times d_i^h}$ on the dofs in $\mathcal{V}^h(D_i)$ and $\mathcal{V}^h(D_j)$. That is, for every entry of A_{ij}^h indexed (l,k), it holds

$$(A_{ij}^h)_{lk} = a_{ij}(\phi_{i,k}^h,\, \phi_{j,l}^h) \quad \text{for } \; l = 1,\dots,d_j^h,\; k = 1,\dots,d_i^h,$$

where $\{\, \phi_{i,k}^h \,\}_{k=1}^{d_i^h}$ denotes the basis of $\mathcal{V}^h(D_i)$. This global matrix is obtained via a standard assembly from local element matrices $A_{\tau,ij}^h$ for the elements $\tau \in \mathcal{T}^h$ formulated on the $\mathcal{V}^h(D_i)$ and $\mathcal{V}^h(D_j)$-dofs associated with τ. The coarse matrices are produced by standard "RAP" procedures. Indeed, the representations of $a_{ij}(\cdot,\cdot)$ in terms of the bases of $\mathcal{V}^H(D_i)$ and $\mathcal{V}^H(D_j)$ is the matrix $A_{ij}^H = P_j^T A_{ij}^h P_i \in \mathbb{R}^{d_j^H \times d_i^H}$. Also, for $T \in \mathcal{T}^H$, using a standard assembly locally with $A_{\tau,ij}^h$ for $\tau \subset T$, the local-on-T fine-scale matrix $A_{T,ij}^h$ is obtained on the $\mathcal{V}^h(D_i)$ and $\mathcal{V}^h(D_j)$-dofs associated with T. Thus, $A_{T,ij}^H = (P_{T,j})^T A_{T,ij}^h P_{T,i}$ forms the coarse element matrices, which can produce A_{ij}^H via a standard assembly. In the case of mixed finite element formulations, where the matrices have a block form, coarse matrices can be obtained either by using the appropriate P_i prolongators for each block separately and combining the results in a coarse block matrix, or equivalently constructing a block diagonal prolongator with the appropriate P_i matrices as diagonal blocks and coarsening the entire block matrix in a monolithic "RAP".

3.2.3. *On the construction of the coarse discrete differential operators and the actions of the coarse cochain projection operators.* Similarly to above, local transition matrices $D_{\tau,i}^h$ are available in terms of the bases and dofs in $\mathcal{V}^h(D_i)$ and $\mathcal{V}^h(D_{i+1})$ for $i = 1,\dots,3$, which produce D_i^h by an overwriting assembly, where $D_{\tau,i}^h$ are effectively submatrices of D_i^h. Also, local-on-T versions $D_{T,i}^h$ can be obtained as needed either via an overwriting assembly or as submatrices of D_i^h. While the procedures constructing coarse bases in ParELAG provide sequences of spaces with desired properties, even in the most basic case the least that can be expected is that D_i^H should map $\mathcal{V}^H(D_i)$ into $\mathcal{V}^H(D_{i+1})$ for $i = 1,\dots,3$. That is, the equality (with a slight abuse of notation) $D_i^H \boldsymbol{v}^H = D_i^h P_i \boldsymbol{v}^H = \Pi_{i+1}^H D_i^h P_i \boldsymbol{v}^H = D_i \boldsymbol{v}^H$ holds for all $\boldsymbol{v}^H \in \mathbb{R}^{d_i^H}$, viewed[4] as

[4]Recall that functions in finite element spaces and algebraic vectors in terms of the respective dofs are identified.

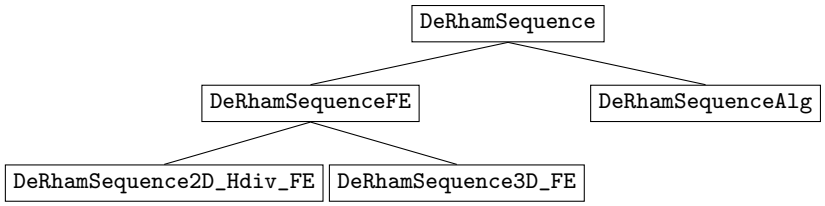


FIGURE 3.4. An illustration of the hierarchy of ParELAG classes for de Rham sequences.

$\boldsymbol{v}^H \in \mathcal{V}^H(D_i)$, where the actions of the operators and the equality signs need to be interpreted in the sense of functions. Note that Π_i^H (see Section 3.3.4) for $i = 1, \ldots, 4$ can be represented as matrices with the dimensions and structure of P_i^T. ParELAG builds the actions of Π_i^H and D_i^H via independent local procedures as the entire coarse bases construction process is local (see Sections 3.2.2 and 3.3, [46, 44, 53, 36]). Observe that the equality $D_i^H = \Pi_{i+1}^H D_i^h P_i$ holds in the sense of matrices; cf. (3.5) below. Moreover, the local actions $\Pi_{T,i}^H$, recognizable as submatrices of Π_i^H on the appropriate dofs, are accessible, and one can obtain $D_{T,i}^H$ as $\Pi_{T,i+1}^H D_{T,i}^h P_{T,i}$ or as a submatrix of D_i^H on the respective dofs.

3.2.4. *The exactness and commutativity properties.* The procedures of constructing the bases (outlined in Section 3.3) and the operators that ParELAG supplies are specially designed to provide the desired properties exactness

$$\text{Range}(D_i^H) = \text{Ker}(D_{i+1}^H) \quad \text{for } i = 1, \ldots, 3 \tag{3.4}$$

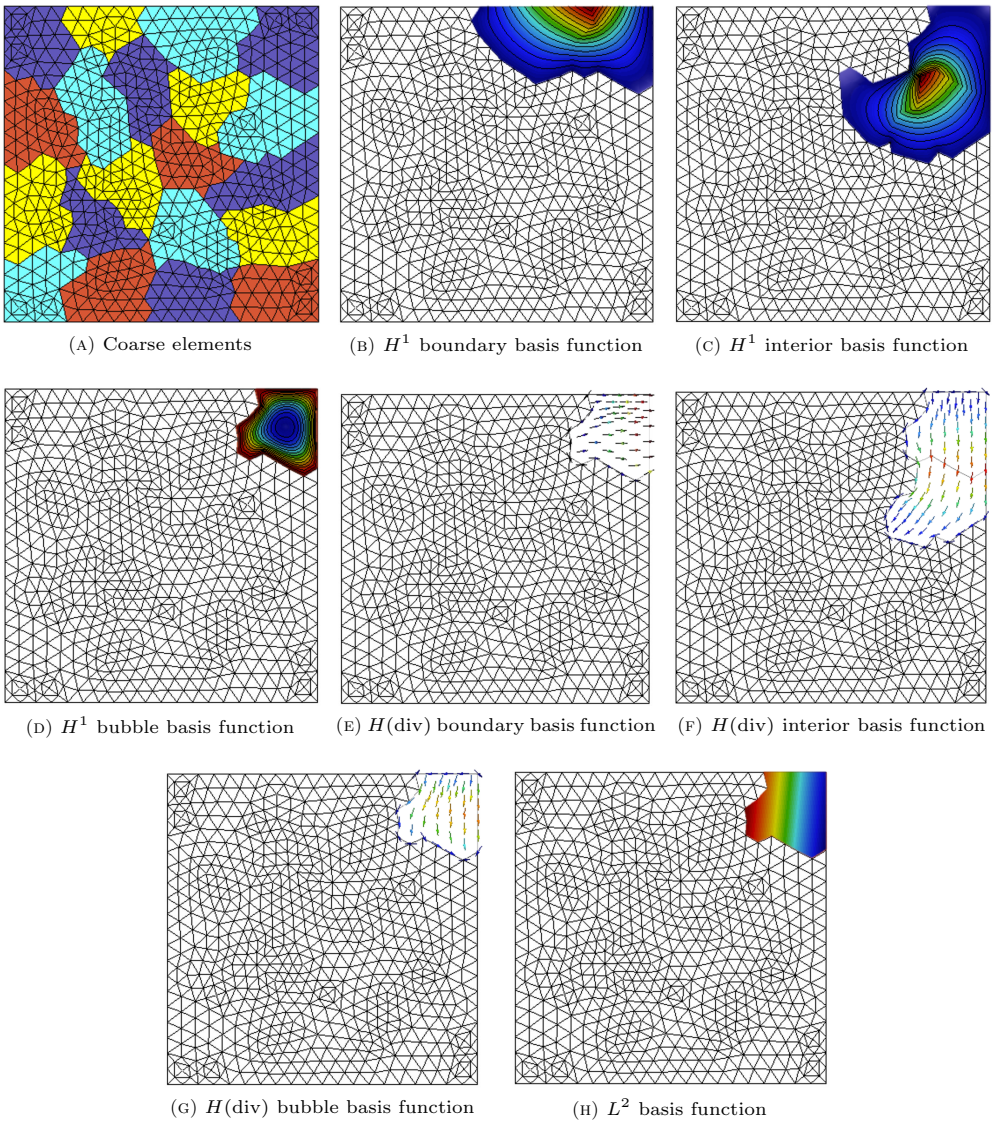
and commutativity (cf. (3.2))

$$D_i^H \circ \Pi_i^H = \Pi_{i+1}^H \circ D_i^h \quad \text{for } i = 1, \ldots, 3. \tag{3.5}$$

It is important to highlight that the obtained coarse spaces and topology of the coarse mesh exhibit fine-like (geometric-like) finite element features. This is a significant property of AMGe utilizing agglomeration of elements, which, together with the algebraic nature of the approach, allows the recursive (cf. Fig. 3.2b) application of the procedures outlined above and in Section 3.3. This provides the capacity of the methodology to supply multilevel hierarchies of nested spaces forming exact and commutative de Rham complexes on all levels.

3.2.5. *On the ParELAG classes for de Rham sequences.* ParELAG delivers a small hierarchy of classes for de Rham sequences. The base class is `DeRhamSequence`, which contains the main toolset necessary for constructing and working with de Rham sequences, including the procedures for building coarse spaces outlined in this paper. Furthermore, it is convenient to have specializations in the form of subclasses for *algebraic* levels (class `DeRhamSequenceAlg`), which are coarse levels produced by ParELAG that are not associated with a given mesh (i.e., that are not *geometric*), as well as for the finest (geometric) level (class `DeRhamSequenceFE`), which is produced employing MFEM. The class `DeRhamSequenceFE` is further specialized to address special cases like the dimensionality of the domain. See Fig. 3.4 for an illustration of the class hierarchy.

3.3. **Coarse bases and the extension procedure.** The abstract construction of coarse basis functions, which is applicable on all levels, is outlined now. The target traces and the extension process are discussed. A detailed presentation of a closely related procedure for coarse space construction can be found in [46].



(A) Coarse elements (B) H^1 boundary basis function (C) H^1 interior basis function

(D) H^1 bubble basis function (E) $H(\text{div})$ boundary basis function (F) $H(\text{div})$ interior basis function

(G) $H(\text{div})$ bubble basis function (H) L^2 basis function

FIGURE 3.5. A two-dimensional illustration of coarse basis functions. Their supports match exactly the respective coarse elements.

3.3.1. *Basics.* The coarse mesh $\mathcal{T}^H$ with all its entities (elements or agglomerates, facets, edges, and vertices) and its topology is available. Note that coarse basis functions are obtained in terms of the fine ones using an extension procedure involving the solution of local finite element problems, which translates into inverting local matrices. This uses information on the association between fine dofs and coarse entities, which is easily derived from the association between fine dofs and the fine entities that constitute each coarse entity. In the terminology of ParELAG, this is called *dof agglomeration* (or *aggregation*), implemented in the `DofAgglomeration` class.

As indicated in Section 3.2.2, the extension can be viewed in the local context of an agglomerate $T \in \mathcal{T}^H$ and its associated lower-dimensional coarse entities. Note that some basis functions are supported on multiple agglomerates. Nevertheless, they are constructed by independent local agglomerate-by-agglomerate processes.

The procedures executed on a single agglomerate T produce the respective shape functions. Shape functions that form a particular basis function agree on fine-scale dofs shared between agglomerates. In the formation of the prolongation matrices, the final basis functions are obtained by joining together all associated shape functions. The shape functions on T alone comprise the local-on-T prolongation matrices, $P_{T,i}$; see Section 3.2.2. Other basis functions are entirely supported in a single agglomerate T. As it is customary, they are called *bubble functions*; see [17]. More specifically, for $i = 1, \dots, 3$, a $\mathcal{V}^h(D_i)$ bubble function in T is nonzero in T, vanishes outside T, and it is globally a function in $H(D_i)$, i.e., its γ_i-trace (see Section 3.1) on ∂T vanishes. To facilitate the discussion of the extension process below, the notion of bubble functions in $\mathcal{V}^h(D_i)$ is used in a slightly extended context to allow the consideration of such functions on lower-dimensional agglomerated entities. This should be intuitively clear and lead to no confusion since the discussion concentrates on local settings associated with particular agglomerated entities. In the case of $\mathcal{V}^h(D_4)$, all coarse basis functions are supported on a single respective agglomerate. Figure 3.5 illustrates coarse basis functions on agglomerates in two dimensions.

3.3.2. *Target traces.* The first step is to select so called targets. The results in this paper are obtained using global polynomial targets due to their simplicity and inherent approximation properties following from standard polynomial approximation theory. Global polynomial targets are set once in ParELAG on the finest level via simple call to the `SetUpscalingTargets()` member function of the `DeRhamSequenceFE` class. The procedure in ParELAG for building the targets is quite simple. Namely, on the finest level, respective monomials, up to a prescribed order, interpolated (i.e., via Π_i^h) on the respective finite element spaces constitute the targets. On coarse levels, the targets are transferred as needed via projection, i.e., by applying the operators Π_i^H, $i = 1, \dots, 4$. Note that it is admissible to utilize polynomial targets of order higher than the order of the finest-level finite elements. In such a case, coarse basis functions are obtained, having a high-order complexion but represented piecewise by lower-order polynomials. Alternatively, local targets on coarse entities can be used, obtained, e.g., via solving local eigenvalue problems, which also provide approximation properties (cf. [36, 18]); or a combination of different targets can be utilized. In any case, for the approach outlined here, appropriate respective target traces on coarse entities (elements or agglomerates, facets, edges, vertices) are obtained and available as needed, represented in terms of respective $\mathcal{V}^h(D_i)$ dofs, $i = 1, \dots, 4$.

The lowest-dimensional sensible traces for $\mathcal{V}^h(D_1), \dots, \mathcal{V}^h(D_4)$, from which extensions are initiated, are respectively on agglomerated vertices, edges, facets, and elements[5]. On these entities, respective separately generated so called *PV traces* (coming from [53]) are included with the (polynomial) targets. These traces alone provide generic lowest-order coarse spaces on $\mathcal{T}^H$. In $\mathcal{V}^h(D_1)$, the PV traces in algebraic form are simply the unity scalar (a vector with a single entry equal to 1) on each agglomerated vertex, while in $\mathcal{V}^h(D_4)$ these are (piecewise) constant functions on each agglomerate. Note that algebraically the latter need not be represented by constant vectors. On the finest level they are obtained via interpolation, Π_4^h, and are typically represented by constant vectors, while on coarse levels they are produced by successive projections, via Π_4^H, and are not represented by constant algebraic vectors.

[5]Note that consequently no extension is needed to construct $\mathcal{V}^H(D_4)$.

In $\mathcal{V}^h(D_2)$ and $\mathcal{V}^h(D_3)$, the PV traces represent the constant unity traces (tangential flow or normal flux) and their associated coarse dofs on the respective agglomerated entities (edges or facets). For example, considering $\mathcal{V}^h(D_3)$, a coarse facet F, and its orientation provided by the vector $\boldsymbol{\varphi}^F$, as described in Section 3.2.1, the normal flux trace on F of the respective PV basis function, $\underline{\phi}^F_{\text{PV},3}$, is the following:

$$\underline{\phi}^F_{\text{PV},3} \cdot \underline{n}_F = \sum_{f \subset F} (\boldsymbol{\varphi}^F)_f \, \underline{\phi}^f_{\text{PV},3} \cdot \underline{n}_f, \text{ satisfying } \int_F \underline{\phi}^F_{\text{PV},3} \cdot \underline{n}_F \, \mathrm{d}\sigma = 1,$$

where $\underline{n}_F$ and $\underline{n}_f$ are the corresponding normal vectors to the coarse facet F and its constituting fine facets, f, which respect their particular orientations, $(\boldsymbol{\varphi}^F)_f$ denotes the vector entry associated with f, and $\underline{\phi}^f_{\text{PV},3}$ are the respective finer-level PV basis functions, which on the finest level are extracted from MFEM. Thus, the algebraic representation of the PV trace on F is derived from the vector $\boldsymbol{\varphi}^F$.

The other target traces on their respective agglomerated entities are othogonalized and made orthogonal to the PV traces in the respective L^2 sense on those entities, removing any linear dependence, using SVD and local mass matrices formulated on the respective trace spaces. Note that this implies that all target traces, apart from the PV traces, have a zero mean and the corresponding trace-space mass matrices produced via "RAP" on all levels, apart from the finest one, are diagonal. This completes the construction of the coarse basis functions for $\mathcal{V}^H(D_4)$, while the rest of the spaces utilize an extension process.

3.3.3. *Extension process.* The extension procedure moves from right to left in the de Rham sequence (3.3) and is quite intricate. A most basic formal outline is provided here and an expository illustration is shown in Fig. 3.6. For a more detailed related discussion, including the feasibility of the extension problems, the demonstration of the exactness (3.4) and commutativity (3.5) properties, and further analysis, see [46].

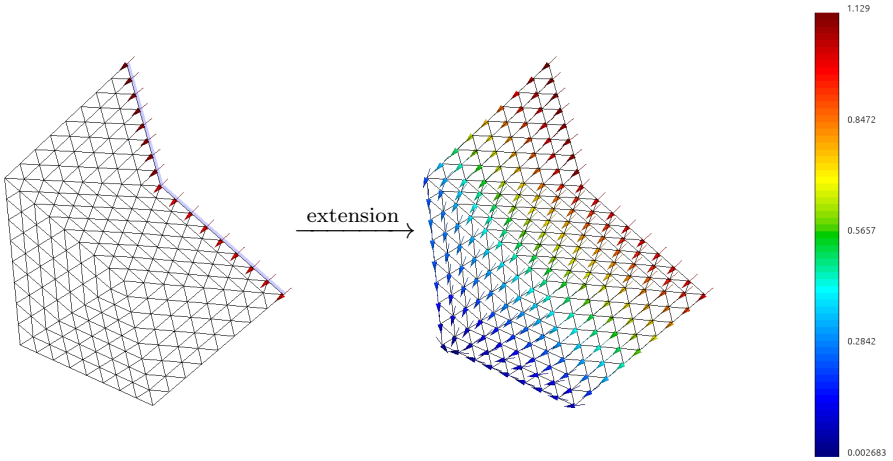
All intricate extension processes outlined here, producing the final interpolation matrices (cf. Section 3.2.2), together with the construction of the coarse cochain projection and discrete differential operators (cf. Sections 3.2.3 and 3.3.4), the initial selection of PV traces and distribution of target traces to agglomerated entities (cf. Section 3.3.2), and all other necessary tasks that compose a complete coarse de Rham sequence are executed via a simple call to the `Coarsen()` member function of the `DeRhamSequence` class. It produces a complete working coarse de Rham sequence, as an object[6] of `DeRhamSequence`, from a current (fine) de Rham sequence.

Extension from the lowest-dimensional traces. The first extension is from the lowest-dimensional, for the respective space, agglomerated entity to a one-dimension-higher agglomerated entity; see Figs. 3.6a and 3.6b for an illustration. That is, for $i = 1, \ldots, 3$, the extensions are respectively vertex to edge, edge to facet, and facet to element. The discussion here is associated with the method `hFacetExtension()` of the class `DeRhamSequence`, called within `DeRhamSequence::Coarsen()`.

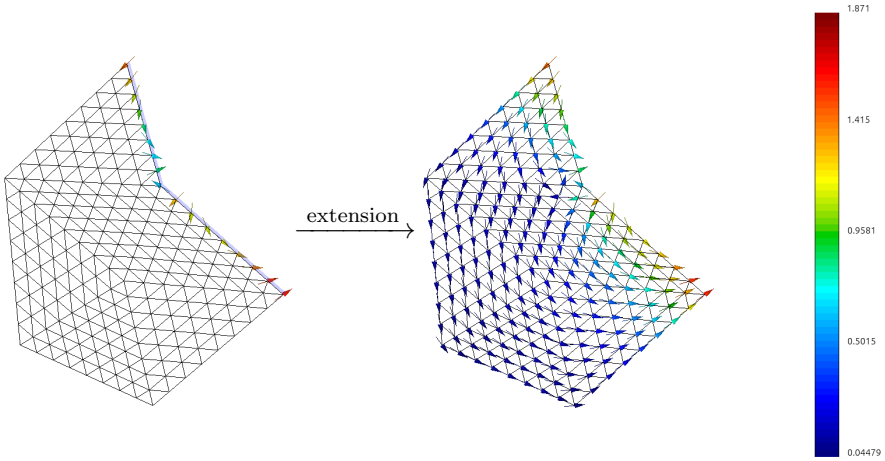
Let L be a lowest-dimensional entity, K a one-dimension-higher entity such that $L \subset \partial K$, and μ a given target trace on L. Using an intuitive abuse of notation, the respective extension of μ to K, ϕ_e, in $\mathcal{V}^h(D_i)$ is obtained by solving the local PDE formally expressed as:

$$\phi_e + D^*_{K,i} \, \psi = 0 \qquad\qquad \text{in } K,$$

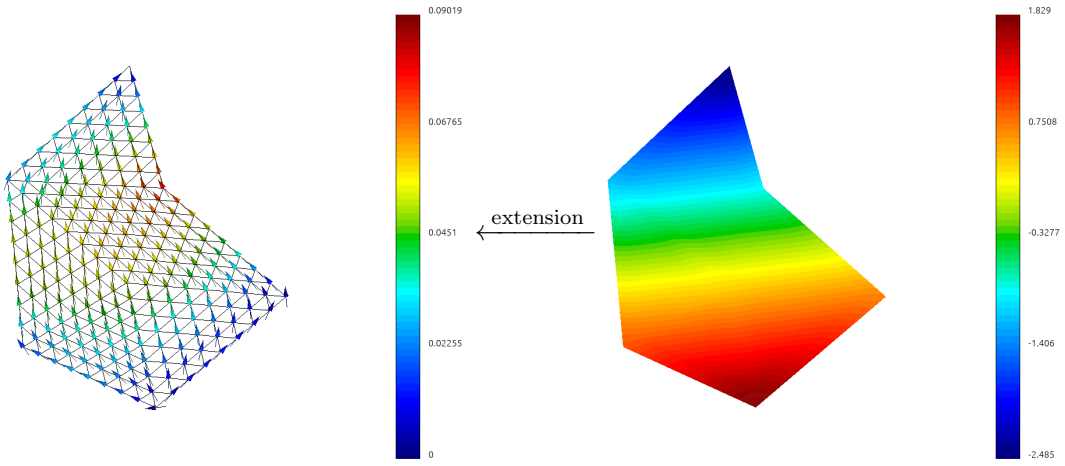
[6]More particularly, as an object of the subclass `DeRhamSequenceAlg` suited for algebraic levels; see Section 3.2.5.



(A) Extension of the PV trace on the agglomerated facet



(B) Extension of a trace, on the agglomerated facet, L^2-orthogonal to the PV trace



(C) Cross-space extension from an L^2 basis function to an $H(\mathrm{div})$ bubble function

FIGURE 3.6. A two-dimensional illustration of local extension procedures producing shape functions in $H(\mathrm{div})$ on a sample agglomerated element, involving respective traces on an agglomerated facet (marked with a light shade) and bubble functions in the agglomerate.

$$\begin{aligned} D_{K,i}\,\phi_e &= c\,\phi^K_{\mathrm{PV},i+1} && \text{in } K,\\ \gamma_i\,\phi_e &= \mu && \text{on } L,\\ \gamma_i\,\phi_e &= 0 && \text{on } \partial K\setminus L, \end{aligned}$$

where $\phi^K_{\mathrm{PV},i+1}$ is the PV trace in $\mathcal{V}^h(D_{i+1})$ associated with K, ψ is a local-on-K function in $\mathcal{V}^h(D_{i+1})$, which is constrained, for solvability, to have a respective zero mean in K, and c is the scalar Lagrangian multiplier associated with that constraint, which represents a scaling, that is a part of the unknowns, providing compatibility of the PDE in terms of properly relating the exterior derivative value, $D_{K,i}\,\phi_e$, and the boundary value, μ, in the Stokes' theorem. Clearly, $c=0$ for all traget traces μ, except when μ represents the respective PV trace in $\mathcal{V}^h(D_i)$ associated with L, i.e., $c=0$ for all zero-mean targets μ on L. Recall that $D_{K,i}$ denotes the differential operator D_i on the entity K, which in the discrete setting, since everything is formulated on the respective local dofs, is represented by a corresponding submatrix of D^h_i, while $D^*_{K,i}$ denotes a formal adjoint, which in the setting of a mixed finite element (weak) discrete formulation is obtained via a corresponding matrix transposition. Note that here the notation γ_i slightly, but intuitively, expands the basic definition of γ_i-traces in Section 3.1. Namely, for $i=1,\dots,3$, L is respectively an agglomerated vertex, edge, and facet, while γ_i on L is respectively a point value on the vertex, tangential flow on the edge, and normal flux on the facet.

Next, $\mathcal{V}^h(D_i)$ bubble functions on K are obtained by a cross-space extension from $\mathcal{V}^h(D_{i+1})$ to $\mathcal{V}^h(D_i)$; see Fig. 3.6c for an illustration.. This assists the procurement of the exactness (3.4). Denote by $\phi^K_{\perp,i+1}$ any L^2-orthogonal to $\phi^K_{\mathrm{PV},i+1}$ (i.e., having a respective zero mean) $\mathcal{V}^h(D_{i+1})$ target trace on K. For each such $\phi^K_{\perp,i+1}$, the corresponding bubble function, ϕ_b, is obtained by solving

$$\begin{aligned} \phi_b + D^*_{K,i}\,\psi &= 0 && \text{in } K,\\ D_{K,i}\,\phi_b &= \phi^K_{\perp,i+1} + c\,\phi^K_{\mathrm{PV},i+1} && \text{in } K,\\ \gamma_i\,\phi_b &= 0 && \text{on } \partial K, \end{aligned}$$

where c is present for stabilizing the system and clearly $c=0$ in the actual solution.

To enhance the approximation properties, further $D_{K,i}$-free[7] bubble functions on K in $\mathcal{V}^h(D_i)$ are produced by projecting the given target traces in $\mathcal{V}^h(D_i)$ associated with K onto the respective space of $D_{K,i}$-free bubble functions and filtering out any linear dependence. In the case of $i=3$, this finishes the process and no further extension is needed.

Further extensions to higher-dimensional agglomerated entities. For the case of $i=2$, one more major extension step (facets $\to$ elements) is necessary, while two such steps (edges $\to$ facets $\to$ elements) are required for $i=1$. Each such step, involving cross-space extensions, has the following generic form. The discussion here, for each extension step, is associated with the method `hRidgePeakExtension()` of the class `DeRhamSequence`, called within `DeRhamSequence::Coarsen()`.

Let N be a lower-dimensional agglomerated entity (but not a lowest-dimensional one), M a one-dimension-higher agglomerated entity such that $N\subset\partial M$, and η a trace on N produced by a previous (lower-dimensional) extension. Observe that at the current stage the construction of $\mathcal{V}^H(D_{i+1})$ is already completed and the nature of the extension procedure, that provided η, makes already known the $\mathcal{V}^H(D_{i+1})$

[7]That is, functions for which applying $D_{K,i}$ gives zero.

function to which D_i would map the final $\mathcal{V}^H(D_i)$ basis function related to η. Thus, appoint the trace on M of that known $\mathcal{V}^H(D_{i+1})$ function, expressed in $\mathcal{V}^h(D_{i+1})$ dofs, as the value of $D_{M,i}\,\eta$, notwithstanding that η is defined only on N. Then, the respective extension of η to M, ϕ_E, in $\mathcal{V}^h(D_i)$ is obtained by solving the following formal local PDE:

$$\begin{aligned} \phi_E + D^*_{M,i}\,\chi &= 0 && \text{in } M,\\ D_{M,i}\,\phi_E - D^*_{M,i+1}D_{M,i+1}\,\chi &= D_{M,i}\,\eta && \text{in } M,\\ \gamma_i\,\phi_E &= \eta && \text{on } N,\\ \gamma_i\,\phi_E &= 0 && \text{on } \partial M \setminus N, \end{aligned}$$

where χ is a local-on-M function in $\mathcal{V}^h(D_{i+1})$.

To facilitate the exactness (3.4), $\mathcal{V}^h(D_i)$ bubble functions on M are produced. During the construction of $\mathcal{V}^H(D_{i+1})$, the basis functions that span $\mathrm{Ker}(D^H_{i+1})$ are known and their traces on M, $\phi^M_{0,i+1}$, are available. Such basis functions are the ones associated with respective D_{i+1}-free bubble functions and target traces with a zero mean (i.e., orthogonal to the respective PV targets). For each such $\phi^M_{0,i+1}$, the corresponding bubble function, ϕ_B, is obtained by solving

$$\begin{aligned} \phi_B + D^*_{M,i}\,\chi &= 0 && \text{in } M,\\ D_{M,i}\,\phi_B - D^*_{M,i+1}D_{M,i+1}\,\chi &= \phi^M_{0,i+1} && \text{in } M,\\ \gamma_i\,\phi_B &= 0 && \text{on } \partial M. \end{aligned}$$

The term $D^*_{M,i+1}D_{M,i+1}\,\chi$ helps to stabilize the system and here it is zero for the final solution.

Finally, the given target traces in $\mathcal{V}^h(D_i)$ associated with M are projected on the space of $D_{M,i}$-free bubble functions on M in $\mathcal{V}^h(D_i)$ and added towards the basis (possibly awaiting further extension) for $\mathcal{V}^H(D_i)$, filtering out any linear dependence. Notice that this is not sensible for $\mathcal{V}^h(D_1)$, since any such bubble function would vanish everywhere. After sufficiently many sweeps (one or two) of the above procedure, the extension process is completed and the coarse de Rham sequence is constructed.

3.3.4. *The coarse cochain projection operators.* Together with the construction of the coarse de Rham sequence, ParELAG builds the projection operators Π^H_i for $i = 1, \ldots, 4$, that are obtainable in sparse matrix form and satisfy the commutativity property (3.5). Their construction parallels the production of the coarse basis functions, starting from the designation of the target traces through moving to higher-dimensional local entities. As indicated in Section 3.2.3, similar to the construction of the prolongation matrices, the projection operators are obtained via independent local agglomerate-by-agglomerate procedures. As a result, on each coarse entity of any admissible dimensionality a local version of the projection operator is obtained from which the global Π^H_i can be assembled by piecing together the local actions, since those actions naturally agree on dofs shared between coarse entities. The independent production of each such local projection operator involves the inversion of a small coarse-scale local mass matrix on the respective coarse entity. More details can be found in [46]; see also [53, 44].

The cochain projection operators do not merely constitute a theoretical tool, but they are also explicitly utilized. Particularly, they are used internally in ParELAG, e.g., in the invocation of AMS and ADS; see Section 4.2. Moreover, they are needed in the implementation of multilevel Monte Carlo methodologies [51, 50] and efficient multilevel nonlinear solvers like FAS (full approximation scheme) [42].

Remark 3.2. The implementation in ParELAG allows, and it is a common practice, the utilization of coefficients in the local extension PDEs. Such coefficients typically come from the particular given problem for which ParELAG is used. For example, the coefficients α and β in (2.1) are incorporated in the appropriate local PDEs. Thus, the extension procedure can be informed about the particular problem of interest and the obtained coarse bases become problem-dependent.

4. Smoothers, coarse solvers, and the multigrid

Generally, multigrid preconditioners, implemented via multilevel cycles, e.g., the well-known V-cycle described by Algorithm 4.1 (see [58]), have as components: a hierarchy of spaces given in the form of a hierarchy of prolongators, a *relaxation* (or *smoothing*) procedure, and a solver or preconditioner for the coarsest problem. The spaces and prolongators are obtained by the processes outlined in Sections 3.2 and 3.3. This section is devoted to describing hybrid ("combined" a.k.a. "Hiptmair") smoothers and coarse AMS and ADS solvers, implemented and utilized in the broad context of generic de Rham sequences as constructed by ParELAG and outlined in Section 3. The hybrid smoothers are implemented within ParELAG, while AMS and ADS are an abstract part of the HYPRE library [1], where ParELAG provides the necessary ingredients, like general transition and projection operators, to utilize them. Note that while in principle Algorithm 4.1 can be used iteratively to obtain a stationary (or fixed-point) iterative method, the main interest here is to apply the preconditioner B_{ML}^{-1} in a preconditioned conjugate gradient (PCG) method for solving problems involving the bilinear forms in (2.1). In that case, Algorithm 4.1 is invoked with $\boldsymbol{x}_0 = \mathbf{0}$.

Algorithm 4.1 A procedure implementing a single multilevel V-cycle. Computes the effect of a multilevel preconditioner B_{ML}^{-1}, i.e., $\boldsymbol{x}_{\text{ML}} = \boldsymbol{x}_0 + B_{\text{ML}}^{-1}(\boldsymbol{b} - A\boldsymbol{x}_0)$.

PROCEDURE: $\boldsymbol{x}_{\text{ML}} \leftarrow \mathbf{ML}\left(A,\ \boldsymbol{b},\ \boldsymbol{x}_0,\ \{M_k\}_{k=1}^{\ell-1},\ \{P_{k+1}^k\}_{k=1}^{\ell-1},\ B_\ell^{-1},\ l\right)$
INPUT: A matrix A, a right-hand side vector $\boldsymbol{b}$, a current iterate $\boldsymbol{x}_0$, a hierarchy of relaxation $\{M_k\}_{k=1}^{\ell-1}$ and prolongation $\{P_{k+1}^k\}_{k=1}^{\ell-1}$ operators, a solver or preconditioner B_ℓ^{-1} on the coarsest level, and a current level l. Here, ℓ is the number of levels in the hierarchy, where a smaller index corresponds to a finer level, and P_{k+1}^k is the prolongator from level $k+1$ to level k. Externally, the procedure is to be invoked on the finest level with a matrix and a right-hand side formulated on the finest level, and with $l = 1$.
OUTPUT: A new multigrid iterate $\boldsymbol{x}_{\text{ML}} \leftarrow \boldsymbol{x}$.
STEPS:
Initialize: $\boldsymbol{x} \leftarrow \boldsymbol{x}_0$.
Pre-relax: $\boldsymbol{x} \leftarrow \boldsymbol{x} + M_l^{-1}\,(\boldsymbol{b} - A\boldsymbol{x})$.
Correct (evoke B_ℓ^{-1} or recurse):
  if $l = \ell - 1$ (i.e., coarsest level reached) **then**
    $\boldsymbol{e}_c \leftarrow B_\ell^{-1}(P_{l+1}^l)^T(\boldsymbol{b} - A\boldsymbol{x})$;
  else
    $\boldsymbol{e}_c \leftarrow \mathbf{ML}\left((P_{l+1}^l)^T A\, P_{l+1}^l,\ (P_{l+1}^l)^T(\boldsymbol{b} - A\boldsymbol{x}),\ \mathbf{0},\ \{M_k\}_{k=1}^{\ell-1},\ \{P_{k+1}^k\}_{k=1}^{\ell-1},\ B_\ell^{-1},\ l+1\right)$;
  end if
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + P_{l+1}^l\,\boldsymbol{e}_c$.
Post-relax: $\boldsymbol{x} \leftarrow \boldsymbol{x} + M_l^{-T}\,(\boldsymbol{b} - A\boldsymbol{x})$.

4.1. Relaxation via hybrid smoothers.

A general level-independent smoothing procedure based on [29] (see also [58, Appendix F]) is outlined here, providing the

relaxation processes $\{M_k\}$ in Algorithm 4.1 for all levels and spaces in the de Rham sequence.

To fix the exposition, consider a generic coarse de Rham complex, using the "H" (superscript) notation, as in (3.3). By the virtue of (2.1), for $i = 1, \dots, 4$, regard bilinear forms $a_{D_i}(u, v) = (\alpha_i D_i u, D_i v)_0 + (\beta_i u, v)_0$ for $u, v \in H(D_i)$ and $\alpha_i > 0$, $\beta_i \geq 0$. In terms of the bases and dofs of $\mathcal{V}^H(D_i)$, these bilinear forms induce matrices $A^H_{D_i}$. With respect to that notation, this work concentrates on the implementation and utilization of Algorithm 4.1 for $A = A^h_{\underline{\text{curl}}}$ and $A = A^h_{\text{div}}$, which arise from the forms in (2.1). Consider given generic (point) smoothers $M^H_{D_i}$ for the respective $A^H_{D_i}$. The "combined" procedure here employs these point smoothers to obtain hybrid smoothers, denoted by $\mathbb{M}^H_{D_i}$, that are used as $\{M_k\}$ in Algorithm 4.1.

Let $\mathbb{M}^H_{D_1} = M^H_{D_1}$, while for $i > 1$, the hybrid smoothers need to "reach" in the reverse direction of the de Rham sequence. Clearly, such a "combined" approach is not necessary for $i = 4$, thus the main utility of these smoothers is for $i = 2, 3$. The procedure is founded upon certain decompositions of the spaces and the exactness in (3.4), similar to the methods in Section 4.2 below. The basic intuitive idea, for $i > 1$, is to regard a decomposition $\mathcal{V}^H(D_i) = \text{Ker}(D^H_i) \oplus [\text{Ker}(D^H_i)]^\perp$ and utilize smoothers that are respectively efficient on $\text{Ker}(D^H_i)$ and $[\text{Ker}(D^H_i)]^\perp$. Smoothing the component in $\text{Ker}(D^H_i)$ is based on (3.4), which allows the utilization of $M^H_{D_{i-1}}$ and D^H_{i-1} as a transition operator, whereas the $[\text{Ker}(D^H_i)]^\perp$ component is addressed by $M^H_{D_i}$. Namely, for $i > 1$ and a given $\boldsymbol{x}_0$, $\boldsymbol{x}_1 = \boldsymbol{x}_0 + (\mathbb{M}^H_{D_i})^{-1}(\boldsymbol{b} - A^H_{D_i}\boldsymbol{x}_0)$ is computed via the following two steps:

$$
\begin{aligned}
\boldsymbol{x}_{\frac{1}{2}} &= \boldsymbol{x}_0 + (M^H_{D_i})^{-1}(\boldsymbol{b} - A^H_{D_i}\boldsymbol{x}_0), \\
\boldsymbol{x}_1 &= \boldsymbol{x}_{\frac{1}{2}} + D^H_{i-1}\,(M^H_{D_{i-1}})^{-1}\,(D^H_{i-1})^T(\boldsymbol{b} - A^H_{D_i}\boldsymbol{x}_{\frac{1}{2}}).
\end{aligned}
\tag{4.1}
$$

That is, the *error propagation* operator satisfies

$$
I - (\mathbb{M}^H_{D_i})^{-1}A^H_{D_i} = \left[I - D^H_{i-1}\,(M^H_{D_{i-1}})^{-1}\,(D^H_{i-1})^T\,A^H_{D_i}\right]\left[I - (M^H_{D_i})^{-1}A^H_{D_i}\right].
$$

Notice that, since generally $D^H_i D^H_{i-1} = 0$, recursively utilizing $\mathbb{M}^H_{D_{i-1}}$ in place of $M^H_{D_{i-1}}$ in (4.1) changes nothing. Therefore, there is no need to "reach" further than one step backwards into the de Rham sequence. To compute an iteration with $(\mathbb{M}^H_{D_i})^{-T}$, reverse the order of the steps in (4.1), while respectively using $(M^H_{D_i})^{-T}$ and $(M^H_{D_{i-1}})^{-T}$ in place of the ones in (4.1).

In general, only $A^H_{D_i}$ may be given and $A^H_{D_{i-1}}$ may not be readily available to derive $M^H_{D_{i-1}}$. In such a case, a practical alternative way is to obtain it "variationally" by setting $A^H_{D_{i-1}} = (D^H_{i-1})^T\,A^H_{D_i}\,D^H_{i-1}$, which is the matrix of a bilinear form on $\mathcal{V}^H(D_{i-1})$ expressed in terms of the dofs and basis in $\mathcal{V}^H(D_{i-1})$. In fact, the respective bilinear form represents a restriction of $a_{D_i}(\cdot,\cdot)$ onto $\text{Ker}(D^H_i)$. That is, the variational $A^H_{D_{i-1}}$ represents $A^H_{D_i}$ on the space $\text{Range}(D^H_{i-1}) = \text{Ker}(D^H_i)$, equipped with the dofs and basis from $\mathcal{V}^H(D_{i-1})$. This is precisely what ParELAG uses.

In this work, the so called ℓ^1-scaled symmetric block Gauss-Seidel smoother [12], as implemented in HYPRE, is used for M_{D_i}. For more details on the analysis of the hybrid approach in a multigrid setting, which counts on the exactness property (3.4), see [58, Appendix F].

4.2. **Coarse solvers using AMS and ADS.** Similarly to Section 4.1, special (a.k.a. regular) decompositions (cf. [32]) of the finite element spaces of interest are used that

allow breaking the problem of obtaining a holistic preconditioner into preconditioning the separate components of the decomposition. This provides an *auxiliary space preconditioner* that reduces the problem to preconditioning a few H^1-type forms, which can be efficiently addressed by AMG, and smoothing. As a part of HYPRE, BoomerAMG is used in this case. In this work, AMS and ADS, possibly wrapped in PCG and performing multiple iterations up to a given tolerance, are to be used as coarse solvers B_ℓ^{-1} in Algorithm 4.1.

For simplicity of exposition, a fine-scale de Rham sequence is considered, using the "h" (superscript) notation, as in (3.1). The utility of such decompositions and methods is general and applicable on generic coarse levels as well, which is of main interest in this work. This is shortly discussed below. Denote the following vector finite element space as $\underline{\mathcal{V}}^h(D_1) = \underline{\mathcal{V}}^h(\underline{\mathrm{grad}}) = [\mathcal{V}^h(\underline{\mathrm{grad}})]^3$ and the following restricted interpolation operators as $\widehat{\Pi}_2^h \colon \underline{\mathcal{V}}^h(\underline{\mathrm{grad}}) \to \mathcal{V}^h(\underline{\mathrm{curl}})$ and $\widehat{\Pi}_3^h \colon \underline{\mathcal{V}}^h(\underline{\mathrm{grad}}) \to \mathcal{V}^h(\mathrm{div})$, where $\widehat{\Pi}_i^h \boldsymbol{r}^h = \Pi_i^h \boldsymbol{r}^h$, in the sense of functions, for $i = 2, 3$ and any $\boldsymbol{r}^h \in \underline{\mathcal{V}}^h(\underline{\mathrm{grad}})$. Notice that $\widehat{\Pi}_i^h$ for $i = 2, 3$ can be viewed as matrices with respect to the corresponding dofs and bases in the discrete spaces. They can be assembled (using overwriting) from locally computed element matrices. Also, the notation in Section 4.1 is utilized.

Consider first the case of $\mathcal{V}^h(\underline{\mathrm{curl}})$ and the auxiliary space preconditioner for $A^h_{\underline{\mathrm{curl}}}$ associated with (2.1). The decomposition of interest is

$$\boldsymbol{v}^h = \tilde{\boldsymbol{v}}^h + \widehat{\Pi}_2^h \boldsymbol{r}^h + D_1^h \boldsymbol{z}^h \quad \text{for } \boldsymbol{v}^h \in \mathcal{V}^h(\underline{\mathrm{curl}}), \tag{4.2}$$

where $\tilde{\boldsymbol{v}}^h \in \mathcal{V}^h(\underline{\mathrm{curl}})$, $\boldsymbol{r}^h \in \underline{\mathcal{V}}^h(\underline{\mathrm{grad}})$, and $\boldsymbol{z}^h \in \mathcal{V}^h(\underline{\mathrm{grad}})$. The exactness and commutativity properties of the sequence, as in (3.4) and (3.5), are instrumental for the existence and stability properties of such decompositions, as shown in [32]. This decomposition inspires an (additive) auxiliary space preconditioner of the following general type:

$$(B^h_{\underline{\mathrm{curl}}})^{-1} = (M^h_{\underline{\mathrm{curl}}})^{-1} + \widehat{\Pi}_2^h \, (\underline{B}^h_{\underline{\mathrm{grad}}})^{-1} \, (\widehat{\Pi}_2^h)^T + D_1^h \, (B^h_{\underline{\mathrm{grad}}})^{-1} \, (D_1^h)^T, \tag{4.3}$$

where $M^h_{\underline{\mathrm{curl}}}$ is a smoother for $A^h_{\underline{\mathrm{curl}}}$, while $\underline{B}^h_{\underline{\mathrm{grad}}}$ is a preconditioner derived from a vector H^1-type formulation and its matrix, $\underline{A}^h_{\underline{\mathrm{grad}}}$, and $B^h_{\underline{\mathrm{grad}}}$ is a preconditioner derived from a scalar H^1-type formulation and its matrix, $A^h_{\underline{\mathrm{grad}}}$, which are typically implemented by invoking AMG. While $\underline{A}^h_{\underline{\mathrm{grad}}}$ and $A^h_{\underline{\mathrm{grad}}}$ can be explicitly provided, assembled from given bilinear forms, ParELAG defaults to utilizing variationally obtained, from the given $A^h_{\underline{\mathrm{curl}}}$, matrices like $\underline{A}^h_{\underline{\mathrm{grad}}} = (\widehat{\Pi}_2^h)^T \, A^h_{\underline{\mathrm{curl}}} \, \widehat{\Pi}_2^h$ and $A^h_{\underline{\mathrm{grad}}} = (D_1^h)^T \, A^h_{\underline{\mathrm{curl}}} \, D_1^h$. The exposition here presents only basic considerations. AMS implements preconditioners like $B^h_{\underline{\mathrm{curl}}}$ and other variations, e.g., multiplicative versions and ones that treat $\underline{\mathcal{V}}^h(\underline{\mathrm{grad}})$ in a scalar component-wise fashion. For more details see [40] and the documentation of HYPRE [1].

Next, consider $\mathcal{V}^h(\mathrm{div})$ and the auxiliary space preconditioner for A^h_{div} associated with (2.1). The decomposition now is

$$\boldsymbol{v}^h = \tilde{\boldsymbol{v}}^h + \widehat{\Pi}_3^h \boldsymbol{r}^h + D_2^h \boldsymbol{z}^h \quad \text{for } \boldsymbol{v}^h \in \mathcal{V}^h(\mathrm{div}), \tag{4.4}$$

where $\tilde{\boldsymbol{v}}^h \in \mathcal{V}^h(\mathrm{div})$, $\boldsymbol{r}^h \in \underline{\mathcal{V}}^h(\underline{\mathrm{grad}})$, and $\boldsymbol{z}^h \in \mathcal{V}^h(\underline{\mathrm{curl}})$. Reusing the above notation in a slightly modified context, the respective (additive) auxiliary space preconditioner is of the following general type:

$$(B^h_{\mathrm{div}})^{-1} = (M^h_{\mathrm{div}})^{-1} + \widehat{\Pi}_3^h \, (\underline{B}^h_{\underline{\mathrm{grad}}})^{-1} \, (\widehat{\Pi}_3^h)^T + D_2^h \, (B^h_{\underline{\mathrm{curl}}})^{-1} \, (D_2^h)^T,$$

where M^h_{div} is a smoother for A^h_{div}, $\underline{B}^h_{\underline{\mathrm{grad}}}$ is a preconditioner for a $\underline{A}^h_{\underline{\mathrm{grad}}}$ matrix (typically, AMG), and $B^h_{\underline{\mathrm{curl}}}$ is a preconditioner like the one in (4.3) for a $A^h_{\underline{\mathrm{curl}}}$ matrix. Notice that in this setting the term $D^h_1 \left(B^h_{\underline{\mathrm{grad}}}\right)^{-1} \left(D^h_1\right)^T$ in (4.3) is dropped, since $D^h_2 D^h_1 = 0$. Again, $\underline{A}^h_{\underline{\mathrm{grad}}}$ and $A^h_{\underline{\mathrm{curl}}}$ can be obtained via assembly from given bilinear forms, including $A^h_{\underline{\mathrm{curl}}}$ can be associated as above with (2.1), but it is convenient to use matrices obtained variationally from the given A^h_{div}. Namely, $\underline{A}^h_{\underline{\mathrm{grad}}} = (\widehat{\Pi}^h_3)^T A^h_{\mathrm{div}} \widehat{\Pi}^h_3$ and $A^h_{\underline{\mathrm{curl}}} = (D^h_2)^T A^h_{\mathrm{div}} D^h_2$. ADS implements preconditioners like B^h_{div} including multiplicative variants and a scalar component-wise approach. Note that B^h_{div} presented here indicates that ADS calls AMS internally as a part of the process. Alternatively, the decomposition (4.2) can be applied in (4.4) to further decompose the component $\boldsymbol{z}^h \in \mathcal{V}^h(\underline{\mathrm{curl}})$, obtaining directly a final decomposition and a respective preconditioner utilizing only smoothers and H^1-type forms. For more details see [41] and the documentation of HYPRE [1].

The utilization of these methods necessitates the provision of matrices D^h_{i-1}, $\widehat{\Pi}^h_i$ or D^H_{i-1}, $\widehat{\Pi}^H_i$ for $i = 2, 3$, where the coarse versions are of interest in this paper. Those are supplied as input parameters to AMS and ADS. Note that D^h_{i-1}, $\widehat{\Pi}^h_i$, and D^H_{i-1} are readily obtainable from MFEM (via an overwriting assembly from element matrices) and the coarsening process in ParELAG, while $\widehat{\Pi}^H_i \colon \underline{\mathcal{V}}^H(\underline{\mathrm{grad}}) \to \mathcal{V}^H(D_i)$ are separately constructed within ParELAG. Namely, considering the setting in (3.3), $\widehat{\Pi}^H_i = \Pi^H_i \, \widehat{\Pi}^h_i \, \underline{P}_1$, where $\underline{P}_1 \colon \underline{\mathcal{V}}^H(\underline{\mathrm{grad}}) \to \underline{\mathcal{V}}^h(\underline{\mathrm{grad}})$ is the interpolation matrix formed as $\underline{P}_1 = \mathrm{diag}(P_1, P_1, P_1)$, while P_1 and Π^H_i are constructed during the coarsening process of Sections 3.2 and 3.3. This is performed recursively to obtain $\widehat{\Pi}^H_i$ on any level.

4.3. **Comments on the solver structures in ParELAG.** The ParELAG library provides access to a variety of solvers for sparse linear systems, including for block systems. Some of them are implemented within the library, like the hybrid smoothers of Section 4.1 and the V-cycle of Algorithm 4.1 for AMGe, while others appropriately setup and invoke external libraries, like HYPRE. Furthermore, these can be properly combined to obtain a final solver method for a linear system of interest.

ParELAG achieves this by first generating a *solver (or preconditioner) library* (an object of class `SolverLibrary`) from a parameter XML file with a very basic and simple syntax. Such a file declares, for a particular computational program, all utilized solvers and preconditioners together with their specific parameters and how they are combined. A solver declaration is essentially assigning a name and a list of parameters for a particular method that ParELAG provides internally. For example, one can declare a solver in the XML file that represents a multigrid method like the one in Algorithm 4.1 and appoint other solvers from the solver library to act as smoothers and coarse solvers, which in turn have their own parameters set up and may internally, as a part of a targeted complex procedure, employ other solvers or preconditioners from the solver library. This is the case when utilizing the methods of this section. A solver library is generated via an invocation of the static member function `CreateLibrary()` of `SolverLibrary`, which builds a `SolverLibrary` object from a provided XML configuration.

To computationally evoke a solver, declared in the XML file, one uses the produced solver library to obtain a respective *solver factory* (an object of class `SolverFactory`), for the particular method being utilized from the ones declared in the solver library, by calling the `GetSolverFactory()` member function of `SolverLibrary`. Such a

factory is applied to construct, via calling the member function `BuildSolver()` of `SolverFactory`, an individual solver (an object of class[8] `Solver`) for a particular given linear system of interest, which then can be invoked to solve the system. The particular solver is invoked following the MFEM style, i.e., by a call to the `Mult()` member in the respective `Solver` class. During the solver construction, the parameters from the XML file are automatically set in the obtained solver, and all necessary related solvers and preconditioners are constructed with the provided parameters using their corresponding factories, which are respectively obtained from the generated solver library. For example, during the construction of a multigrid preconditioner, the respective smoothers and the solver on the coarsest level are built as a part of the process of manufacturing the multigrid preconditioner. Such a paradigm of solver structuring may sound familiar to a reader that has been exposed to some of the popular available solver libraries.

Respective parameter XML files, employing and combining solvers relevant to this work and outlined in this paper, and their utilization are demonstrated with the ParELAG mini applications in MFEM. Generic solver factories for AMS, ADS, Krylov space methods, hybrid smoothers, AMGe cycles, block preconditioners, other methods within HYPRE and MFEM, etc. are implemented and available in ParELAG, and they are accessible for use in parameter XML files.

5. Numerical examples

This section contains numerical results employing ParELAG in the context of the discussed AMGe multigrid solvers for (2.1). The numerical examples are preceded by a short comment on the ParELAG mini applications in MFEM, also used for computing the results, and a description of the sample problem.

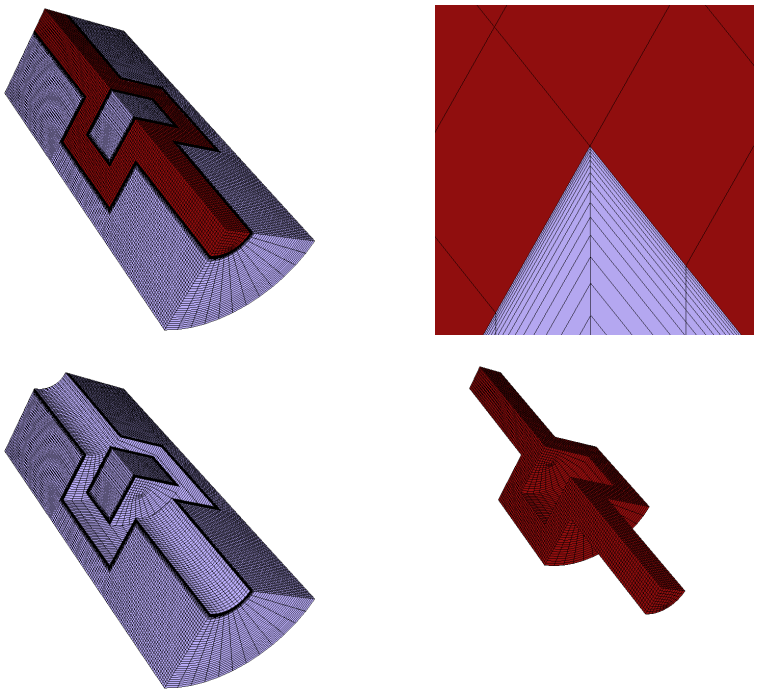
5.1. **On the mini applications.**

A brief comment on the demonstrative ParELAG mini applications [4] in MFEM is presented here. Two respective mini applications are provided withing MFEM together with corresponding sample XML parameter files (cf. Section 4.3) – one mini application for the $H(\underline{\text{curl}})$ case and another for $H(\text{div})$. While for simplicity and clarity they are separate applications, their general structure is analogous as summarized below.

After loading and refining, as appointed, a given mesh file, the applications follow the overall procedure outlined above; cf. Sections 3.2 and 3.3. Namely, successively the coarse meshes and topologies are constructed (Sections 3.2.1 and 3.3.1) by reverting MFEM-performed geometric refinements (utilizing `MFEMRefinedMeshPartitioner`), coarse polynomial traces are identified (Section 3.3.2), and coarse spaces are constructed by producing their basis functions via the intricate extension procedure (Section 3.3.3), including the construction of prolongation matrices (Section 3.2.2), and coarse discrete differential operators and cochain projectors (Sections 3.2.3 and 3.3.4). All these are achieved via simple invocations of ParELAG. Moreover, the matrices for the respective bilinear forms (2.1) are obtained on all levels (cf. Section 3.2.2). Note that boundary conditions are imposed through standard elimination of the respective dofs, which is performed on all levels, including the coarse[9] ones, since respective boundary dofs are distinguishable on all levels (cf. the last paragraph of Section 3.2.1).

[8]The base `Solver` class is declared within MFEM.

[9]In more detail, in the case here, coarse matrices are obtained via "RAP" within the solver portion of ParELAG, where the boundary dofs are directly eliminated in the prolongation matrices.



$(\alpha, \beta) = (1.641, 0.2)$ $(\alpha, \beta) = (0.00188, 2000)$

FIGURE 5.1. Domain, initial or starting mesh (including a close-up of the graded mesh in the upper right) of 116640 hexahedral elements, and piecewise constant coefficients for the numerical examples.

In the end, the solver of interest, using PCG preconditioned by an AMGe V-cycle with the respective smoothers and coarse solvers as described in Section 4, is invoked to solve the corresponding fine-level system associated with (2.1). Note that the solver and the particularities concerning it are entirely formulated and selected in the respective XML parameter files (Section 4.3) and there are no solver specific constructs indicated in the code of the mini applications.

5.2. **On the experiments settings.** The particular test setting is inspired by a "crooked pipe" problem, which involves scalar coefficients with large jumps and a graded mesh with highly stretched (anisotropic) elements. For demonstration, the methodology is applied for solving linear systems coming from discretizations of formulations using the bilinear forms in (2.1), a constant right-hand side, and homogeneous essential boundary conditions for simplicity. The utilized domain, (coarsest) mesh, and piecewise constant coefficients are depicted in Fig. 5.1, where $(\alpha, \beta) = (1.641, 0.2)$ in the lighter portion of the domain and $(\alpha, \beta) = (0.00188, 2000)$ in the darker portion.

The systems coming from (2.1) are solved using PCG preconditioned by a single AMGe V-cycle; see Section 4. The iterative process is stopped when the relative size of the residual, measured by the preconditioner-induced norm, is reduced by 10^{-6}. Note that the settings in the XML files take the relative reduction of the norm as a parameter in the criteria, i.e., the parameters are set to 10^{-6}, whereas the inner solver displays and works with the respective inner products (without square roots), making the utilized tolerance appear and be automatically internally squared to 10^{-12}.

# refs	# procs	# elems	ℓ	$H(\underline{\text{curl}})$			$H(\text{div})$		
				# dofs	# it_e	# it_a	# dofs	# it_e	# it_a
2	72	7,464,960	3	22,772,484	131	93	22,583,232	36	31
3	576	59,719,680	3	180,667,656	198	118	179,912,448	55	44
4	4,608	477,757,440	3	1,439,303,184	231	148	1,436,285,952	86	60
3	576	59,719,680	4	180,667,656	169	118	179,912,448	54	44
4	4,608	477,757,440	5	1,439,303,184	190	148	1,436,285,952	77	60

TABLE 5.1. Solver results with lowest order finite elements for both $H(\underline{\text{curl}})$ and $H(\text{div})$, as provided by (2.1) and Fig. 5.1. In all cases, # elems / # procs = 103,680 on the finest level.

The V-cycles use AMGe hierarchies, as described above, and hybrid smoothers for pre and post-relaxation; see Section 4.1. An application of the hybrid smoother invokes two sweeps of ℓ^1-scaled symmetric block Gauss-Seidel for each *primary* and *auxiliary* smoothings within the hybrid approach. A fixed number of five iterations of PCG preconditioned by AMS or ADS, respectively, serves as a solver on the coarsest level[10]; see Section 4.2.

In the tests, the mesh in Fig. 5.1 is refined to obtain a fine-grid problem, which is consequently solved in parallel by the methods discussed in this paper. Experiments are performed in a *weak scaling* setting, in the sense that as the mesh is refined uniformly, the number of processors is increased accordingly so that the number of elements per processor is maintained constant.

5.3. **Results.** Computational results on solving systems induced by (2.1) in parallel for these generally challenging problems are presented here, employing lowest and next to the lowest order finite elements and uniformly refining the initial mesh in Fig. 5.1. The tests utilize the Quartz cluster at Lawrence Livermore National Laboratory. It is equipped on each node with 128 GB of memory and two 18-core Intel Xeon E5-2695 v4 CPUs at 2.1 GHz, resulting in 36 computational cores per node, and the total number of computational nodes (cores) is 2,988 (107,568). The peak single CPU memory bandwidth is 77 GB/s and the Cornelis Networks Omni-Path provides the inter-node connection.

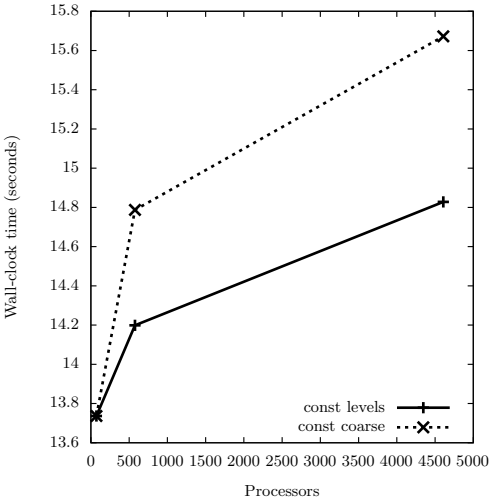
The number of PCG, using AMGe, iterations (# it_e), the number of dofs (# dofs), and the number of elements (# elems) on the finest level are reported, as well as the number of uniform mesh refinements (# refs) employed to obtain the fine mesh, the total number of levels (denoted by ℓ) in the AMGe hierarchy, and the number of utilized processors[11] (# procs).

For completeness, respective results with AMS and ADS are provided as state of the art methods. This includes the number of PCG, using AMS or ADS, iterations (# it_a) and wall-clock timings.

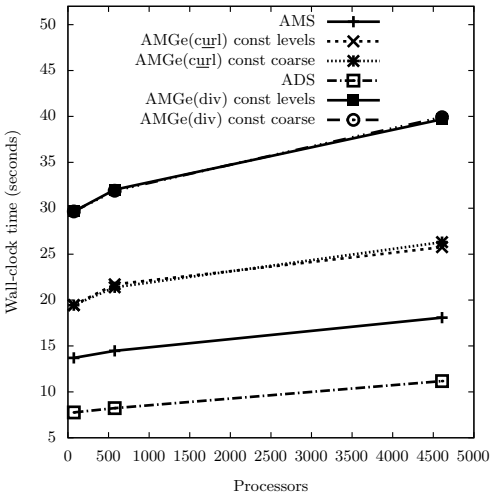
5.3.1. *Results for lowest order elements.* Problem information and solvers iterations are shown in Table 5.1 for both $H(\underline{\text{curl}})$ and $H(\text{div})$. Two test cases are demonstrated: one where the number of AMGe levels is kept fixed (equal to 3) as the fine mesh is refined, essentially also refining the coarsest level, and another where as the fine

[10]The particular choice is largely motivated by the objective to demonstrate the functionality of the library and the ability to combine a variety of solvers and smoothers in the XML files to obtain a sophisticated combined final method. Using a singe or a few applications of the respective AMS or ADS, without PCG, is also a valid option here.

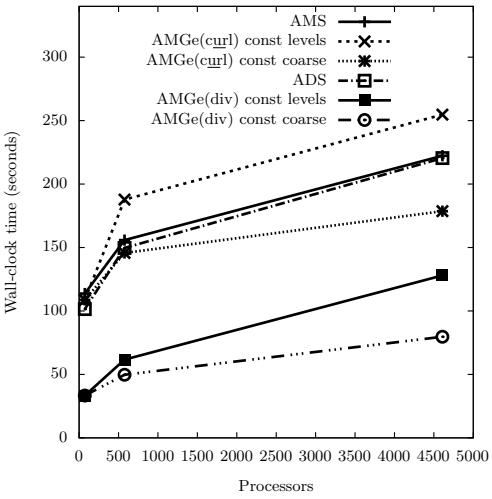
[11]Strictly speaking, this is the number of individual independent computational units, i.e., cores.



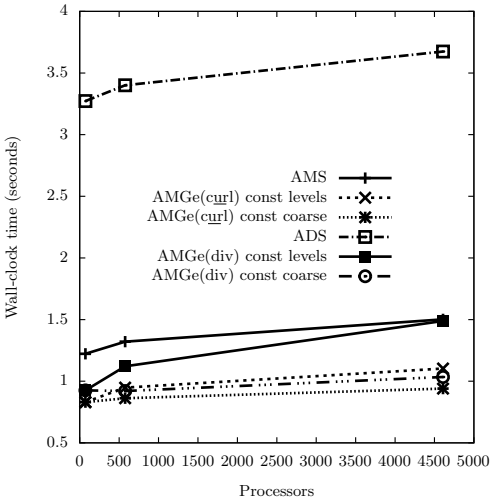
(A) Coarse de Rham sequences build



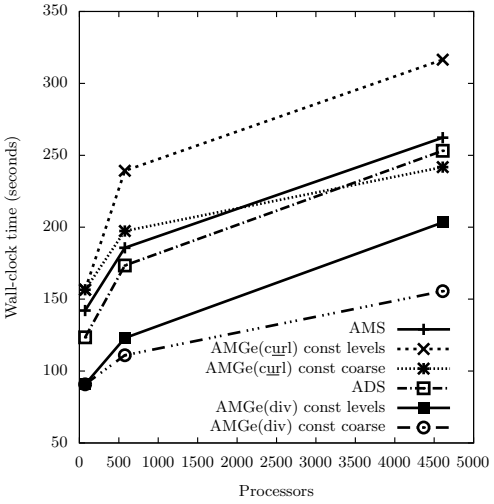
(B) Solver initialization



(C) Total solve time



(D) Average time per iteration



(E) Total program execution

FIGURE 5.2. Weak scaling with lowest order finite elements, where # elems / # procs = 103,680 on the finest level.

# refs	# procs	# elems	ℓ	$H(\underline{\mathrm{curl}})$			$H(\mathrm{div})$		
				# dofs	# $\mathrm{it_e}$	# $\mathrm{it_a}$	# dofs	# $\mathrm{it_e}$	# $\mathrm{it_a}$
2	720	7,464,960	3	180,667,656	191	166	179,912,448	60	57
3	5,760	59,719,680	3	1,439,303,184	269	223	1,436,285,952	100	78
3	5,760	59,719,680	4	1,439,303,184	285	223	1,436,285,952	84	78

TABLE 5.2. Solver results with next to the lowest order finite elements for both $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$, as provided by (2.1) and Fig. 5.1. In all cases, # elems / # procs = 10,368 on the finest level.

mesh is refined the number of levels is increased so that the coarsest level is constant and coinciding with the initial mesh presented in Fig. 5.1.

Timing plots, using wall-clock times as reported by the code of the miniapps, are shown in Fig. 5.2, including wall-clock times for the entire program executions. Clearly, AMGe($\underline{\mathrm{curl}}$) and AMGe(div) denote the AMGe solvers for the $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$-conforming problems, respectively, and both cases of constant number of levels and a constant size of the coarsest problem are shown. The construction of the whole coarse de Rham sequences is also reported, which includes the element agglomeration times, the local extension procedures, and building other necessary constructs. Notice that the construction time does not grow much, especially in the case of constant number of levels, since the majority of the spent cost is on the local extension procedures, which scale perfectly, as they involve no communication.

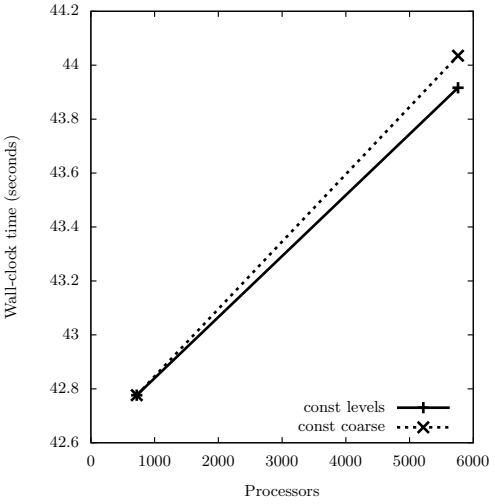
Observe that the AMGe approach performs well and is comparable to the state of the art represented by AMS and ADS. Quite interestingly, Fig. 5.2 indicates that the case of increasing the number of AMGe levels demonstrates better scalability. To exploit this scalability potential in practice for extremely large problems in the setting of extreme parallelism, parallel redistribution and load balancing would be needed on coarse levels obtained via AMGe to allow sufficient coarsening when large number of processors are utilized. This is a subject of an ongoing work.

5.3.2. *Results for next to the lowest order elements.* For completeness, results using next to the lowest order finite elements are presented in Table 5.2 and Fig. 5.3, following the paradigm of the previous subsection. Again, the AMGe methodology performs well and is comparable to the state of the art.

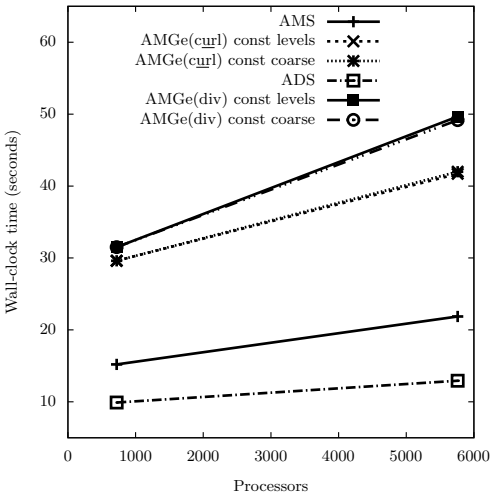
Note that the finest MFEM-generated level and the coarse ParELAG-generated levels in the miniapps by default interpret *next to the lowest order* slightly differently, even if similar, especially when employing hexahedral elements. Consider for example the L^2-conforming finite elements spaces of piecewise polynomial functions. Being informed about the geometry of the elements and their tensor-product structure, MFEM produces bilinear elements with 8 dofs and basis functions per element. In contrast, ParELAG operates in a generic geometry-agnostic way and by default the finite element order determines the order of the targets. Thus, by default it produces targets and spaces that provide piecewise linear interpolation independently of the shape of the element, resulting in 4 dofs and basis functions per element. Generally, this behavior can be easily altered by changing the way targets are selected, but the concentration here is on the default behavior.

6. Conclusions and future work

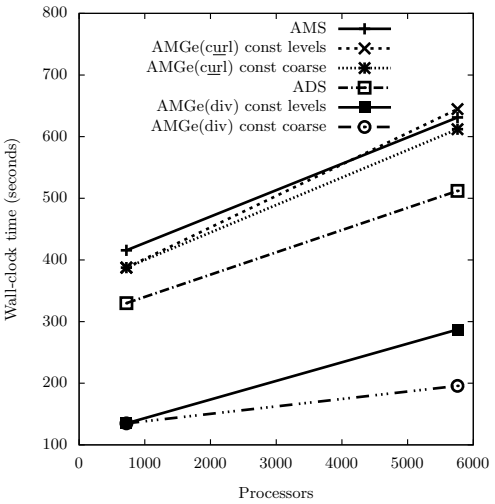
In this paper, we have introduced an AMGe approach for $H(\underline{\mathrm{curl}})$ and $H(\mathrm{div})$ formulations. It involves coarsening of the de Rham sequence, utilizing hybrid smoothers, and evoking the current state-of-the-art solvers, AMS and ADS, for



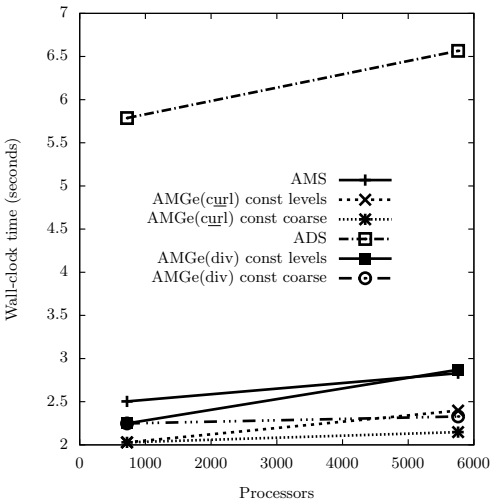
(A) Coarse de Rham sequences build



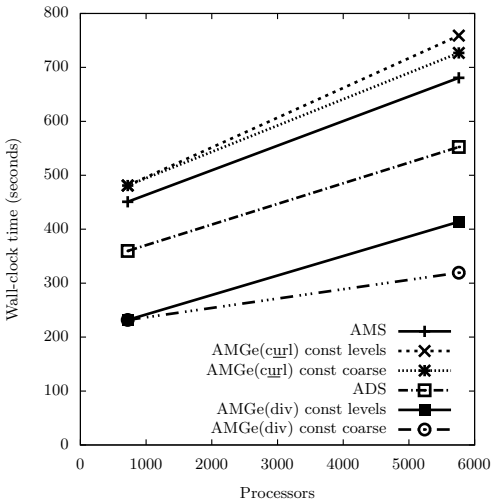
(B) Solver initialization



(C) Total solve time



(D) Average time per iteration



(E) Total program execution

FIGURE 5.3. Weak scaling with next to the lowest order finite elements, where # elems / # procs = 10,368 on the finest level.

solving the coarsest problems. The methods are described and their performance shown in numerical examples. Also, an overview of ParELAG is presented. The approach demonstrates good performance for problems of interest and partially reveals the utility of ParELAG. The library has capacity and potential to be useful for addressing a variety of other problems and settings. Currently, ParELAG does not admit agglomerated elements that are shared or redistributed between processors. This is a potential limitation of the scalability of the library. The development of such functionality is a currently ongoing work. Also, the hybridization and static condensation methods of [24, 38, 37] can be implemented for solving the coarsest $H(\text{div})$ problems, in lieu of invoking ADS.

References

[1] HYPRE: Scalable Linear Solvers and Multigrid Methods. http://computing.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods.

[2] METIS: Graph Partitioning and Fill-reducing Matrix Ordering. http://glaros.dtc.umn.edu/gkhome/views/metis.

[3] MFEM: Modular Finite Element Methods Library. http://mfem.org. doi:10.11578/dc.20171025.1248.

[4] ParELAG mini applications in MFEM. https://github.com/mfem/mfem/tree/master/miniapps/parelag.

[5] ParELAG: Parallel Element Agglomeration Algebraic Multigrid Upscaling and Solvers. http://github.com/LLNL/parelag.

[6] J H Adler and P S Vassilevski. Improving Conservation for First-Order System Least-Squares Finite-Element Methods. In Oleg P Iliev, Svetozar D Margenov, Peter D Minev, Panayot S Vassilevski, and Ludmil T Zikatanov, editors, *Numer. Solut. Partial Differ. Equations Theory, Algorithms, Their Appl.*, pages 1–19, 2013. doi:10.1007/978-1-4614-7172-1_1.

[7] JH Adler and Panayot S Vassilevski. Error analysis for constrained first-order system least-squares finite-element methods. *SIAM Journal on Scientific Computing*, 36(3):A1071–A1088, 2014.

[8] Douglas N Arnold, Richard S Falk, and Jay Gopalakrishnan. Mixed Finite Element Approximation of the Vector Laplace with Dirichlet Boundary Conditions. *Math. Model. Methods Appl. Sci.*, 22(09):1250024, 2012. doi:10.1142/S0218202512500248.

[9] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Preconditioning in $\mathbf{H}(\text{div})$ and applications. *Math. Comput.*, 66(219):957–985, 1997. doi:10.1090/S0025-5718-97-00826-0.

[10] Douglas N Arnold, Richard S Falk, and Ragnar Winther. Multigrid in $H(\text{div})$ and $H(\text{curl})$. *Numer. Math.*, 85(2):197–217, 2000. doi:10.1007/PL00005386.

[11] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Finite element exterior calculus: from Hodge theory to numerical stability. *Bull. Am. Math. Soc.*, 47(2):281–354, 2010. doi:10.1090/S0273-0979-10-01278-4.

[12] A Baker, R Falgout, T Kolev, and U Yang. Multigrid Smoothers for Ultraparallel Computing. *SIAM J. Sci. Comput.*, 33(5):2864–2887, 2011. doi:10.1137/100798806.

[13] Nathan Bell and Luke N Olson. Algebraic multigrid for k-form Laplacians. *Numer. Linear Algebr. with Appl.*, 15(2-3):165–185, 2008. doi:10.1002/nla.577.

[14] Pavel B Bochev, Christopher J Garasi, Jonathan J Hu, Allen C Robinson, and Raymond S Tuminaro. An Improved Algebraic Multigrid Method for Solving Maxwell's Equations. *SIAM J. Sci. Comput.*, 25(2):623–642, 2003. doi:10.1137/S1064827502407706.

[15] Pavel B Bochev, Jonathan J Hu, Allen C Robinson, and Raymond S Tuminaro. Towards robust 3D Z-pinch simulations: Discretization and fast solvers for magnetic diffusion in heterogeneous conductors. *Electron. Trans. Numer. Anal.*, 15:186–210, 2003.

[16] Pavel B Bochev, Jonathan J Hu, Christopher M Siefert, and Raymond S Tuminaro. An Algebraic Multigrid Approach Based on a Compatible Gauge Reformulation of Maxwell's Equations. *SIAM J. Sci. Comput.*, 31(1):557–583, 2008. doi:10.1137/070685932.

[17] Daniele Boffi, Franco Brezzi, and Michel Fortin. *Mixed Finite Element Methods and Applications*, volume 44 of *Springer Series in Computational Mathematics*. Springer, Berlin, Heidelberg, 2013.

[18] Marian Brezina and Panayot S Vassilevski. Smoothed Aggregation Spectral Element Agglomeration AMG: SA-ρAMGe. In Ivan Lirkov, Svetozar Margenov, and Jerzy Waśniewski, editors, *Large-Scale Sci. Comput.*, pages 3–15, Berlin, Heidelberg, 2012. Springer.

[19] Thomas A Brunner. Forms of Approximate Radiation Transport. Technical report, SAND2002-1778, Sandia National Laboratories, 2002. doi:10.2172/800993.

[20] Z Cai, R Lazarov, T A Manteuffel, and S F McCormick. First-Order System Least Squares for Second-Order Partial Differential Equations: Part I. *SIAM J. Numer. Anal.*, 31(6):1785–1799, 1994. doi:10.1137/0731091.

[21] Zhiqiang Cai, Charles Tong, Panayot S Vassilevski, and Chunbo Wang. Mixed finite element methods for incompressible flow: Stationary Stokes equations. *Numer. Methods Partial Differ. Equ.*, 26(4):957–978, 2010. doi:10.1002/num.20467.

[22] Zhiqiang Cai, Chunbo Wang, and Shun Zhang. Mixed Finite Element Methods for Incompressible Flow: Stationary Navier-Stokes Equations. *SIAM J. Numer. Anal.*, 48(1):79–94, 2010. doi:10.1137/080718413.

[23] T Chartier, R Falgout, V Henson, J Jones, T Manteuffel, S McCormick, J Ruge, and P Vassilevski. Spectral AMGe (ρAMGe). *SIAM J. Sci. Comput.*, 25(1):1–26, 2003. doi:10.1137/S106482750139892X.

[24] V Dobrev, T Kolev, C S Lee, V Tomov, and P S Vassilevski. Algebraic Hybridization and Static Condensation with Application to Scalable $H(\mathrm{div})$ Preconditioning. *SIAM J. Sci. Comput.*, 41(3):B425–B447, 2019. doi:10.1137/17M1132562.

[25] Hillary R Fairbanks, Sarah Osborn, and Panayot S Vassilevski. Estimating posterior quantity of interest expectations in a multilevel scalable framework. *Numerical Linear Algebra with Applications*, page e2352, 2020.

[26] H.R. Fairbanks, U. Villa, and P.S. Vassilevski. Multilevel hierarchical decomposition of finite element white noise with application to multilevel markov chain monte carlo. *SIAM Journal on Scientific Computing*, in press. arXiv:2007.14440.

[27] Robert D Falgout and Panayot S Vassilevski. On Generalizing the Algebraic Multigrid Framework. *SIAM J. Numer. Anal.*, 42(4):1669–1693, 2004. doi:10.1137/S0036142903429742.

[28] R Hiptmair. Multigrid method for $H(\mathrm{div})$ in three dimensions. *Electron. Trans. Numer. Anal.*, 6:133–152, 1997.

[29] R Hiptmair. Multigrid Method for Maxwell's Equations. *SIAM J. Numer. Anal.*, 36(1):204–225, 1998. doi:10.1137/S0036142997326203.

[30] R Hiptmair. Finite elements in computational electromagnetism. *Acta Numer.*, 11:237–339, 2002. doi:10.1017/S0962492902000041.

[31] R Hiptmair, G Widmer, and J Zou. Auxiliary space preconditioning in $H_0(\mathrm{curl};\Omega)$. *Numer. Math.*, 103(3):435–459, 2006. doi:10.1007/s00211-006-0683-0.

[32] R Hiptmair and J Xu. Nodal Auxiliary Space Preconditioning in $\mathbf{H}(\mathbf{curl})$ and $\mathbf{H}(\mathrm{div})$ Spaces. *SIAM J. Numer. Anal.*, 45(6):2483–2509, 2007. doi:10.1137/060660588.

[33] Ralf Hiptmair and Andrea Toselli. Overlapping and Multilevel Schwarz Methods for Vector Valued Elliptic Problems in Three Dimensions. In Petter Bjørstad and Mitchell Luskin, editors, *Parallel Solut. Partial Differ. Equations*, pages 181–208, 2000. doi:10.1007/978-1-4612-1176-1_8.

[34] J Jones and B Lee. A Multigrid Method for Variable Coefficient Maxwell's Equations. *SIAM J. Sci. Comput.*, 27(5):1689–1708, 2006. doi:10.1137/040608283.

[35] Jim E Jones and Panayot S Vassilevski. AMGe Based on Element Agglomeration. *SIAM J. Sci. Comput.*, 23(1):109–133, 2001. doi:10.1137/S1064827599361047.

[36] D Z Kalchev, C S Lee, U Villa, Y Efendiev, and P S Vassilevski. Upscaling of Mixed Finite Element Discretization Problems by the Spectral AMGe Method. *SIAM J. Sci. Comput.*, 38(5):A2912–A2933, 2016. doi:10.1137/15M1036683.

[37] Delyan Z Kalchev and Panayot Vassilevski. A Condensed Constrained Nonconforming Mortar-Based Approach for Preconditioning Finite Element Discretization Problems. *SIAM J. Sci. Comput.*, 42(5):A3136–A3156, 2020. doi:10.1137/19M1305690.

[38] Delyan Z Kalchev and Panayot S Vassilevski. Auxiliary Space Preconditioning of Finite Element Equations Using a Nonconforming Interior Penalty Reformulation and Static Condensation. *SIAM J. Sci. Comput.*, 42(3):A1741–A1764, 2020. doi:10.1137/19M1286815.

[39] Tzanio V Kolev, Joseph E Pasciak, and Panayot S Vassilevski. $\mathbf{H}(\mathrm{curl})$ auxiliary mesh preconditioning. *Numer. Linear Algebr. with Appl.*, 15(5):455–471, 2008. doi:10.1002/nla.534.

[40] Tzanio V Kolev and Panayot S Vassilevski. Parallel Auxiliary Space AMG for $H(\mathrm{curl})$ Problems. *J. Comput. Math.*, 27(5):604–623, 2009. doi:10.4208/jcm.2009.27.5.013.

[41] Tzanio V Kolev and Panayot S Vassilevski. Parallel Auxiliary Space AMG Solver for $H(\text{div})$ Problems. *SIAM J. Sci. Comput.*, 34(6):A3079–A3098, 2012. doi:10.1137/110859361.

[42] Max la Cour Christensen, Panayot S Vassilevski, and Umberto Villa. Nonlinear multigrid solvers exploiting AMGe coarse spaces with approximation properties. *J. Comput. Appl. Math.*, 340:691–708, 2018. doi:10.1016/j.cam.2017.10.029.

[43] Max la Cour Christensen, Umberto Villa, Allan P Engsig-Karup, and Panayot S Vassilevski. Numerical Multilevel Upscaling for Incompressible Flow in Reservoir Simulation: An Element-Based Algebraic Multigrid (AMGe) Approach. *SIAM J. Sci. Comput.*, 39(1):B102–B137, 2017. doi:10.1137/140988991.

[44] I V Lashuk and P S Vassilevski. Element agglomeration coarse Raviart-Thomas spaces with improved approximation properties. *Numer. Linear Algebr. with Appl.*, 19(2):414–426, 2012. doi:10.1002/nla.1819.

[45] Ilya Lashuk and Panayot S Vassilevski. On some versions of the element agglomeration AMGe method. *Numer. Linear Algebr. with Appl.*, 15(7):595–620, 2008. doi:10.1002/nla.585.

[46] Ilya V Lashuk and Panayot S Vassilevski. The Construction of the Coarse de Rham Complexes with Improved Approximation Properties. *Comput. Methods Appl. Math.*, 14(2):257–303, 2014. doi:10.1515/cmam-2014-0004.

[47] Ping Lin. A Sequential Regularization Method for Time-Dependent Incompressible Navier-Stokes Equations. *SIAM J. Numer. Anal.*, 34(3):1051–1071, 1997. doi:10.1137/S0036142994270521.

[48] Peter Monk. *Finite Element Methods for Maxwell's Equations*. Numerical Mathematics and Scientific Computation. Clarendon Press, Oxford, 2003.

[49] Duk-Soon Oh, Olof B Widlund, Stefano Zampini, and Clark R Dohrmann. BDDC Algorithms with deluxe scaling and adaptive selection of primal constraints for Raviart-Thomas vector fields. *Math. Comput.*, 87(310):659–692, 2018. doi:10.1090/mcom/3254.

[50] Sarah Osborn, Panayot S Vassilevski, and Umberto Villa. A Multilevel, Hierarchical Sampling Technique for Spatially Correlated Random Fields. *SIAM J. Sci. Comput.*, 39(5):S543–S562, 2017. doi:10.1137/16M1082688.

[51] Sarah Osborn, Patrick Zulian, Thomas Benson, Umberto Villa, Rolf Krause, and Panayot S Vassilevski. Scalable hierarchical PDE sampler for generating spatially correlated random fields using nonmatching meshes. *Numer. Linear Algebr. with Appl.*, 25(3):e2146, 2018. doi:10.1002/nla.2146.

[52] J E Pasciak and J Zhao. Overlapping Schwarz methods in $\mathbf{H}(\mathbf{curl})$ on polyhedral domains. *J. Numer. Math.*, 10(3):221–234, 2002. doi:10.1515/JNMA.2002.221.

[53] Joseph E Pasciak and Panayot S Vassilevski. Exact de Rham Sequences of Spaces Defined on Macro-Elements in Two and Three Spatial Dimensions. *SIAM J. Sci. Comput.*, 30(5):2427–2446, 2008. doi:10.1137/070698178.

[54] A I Pehlivanov, G F Carey, and P S Vassilevski. Least-squares mixed finite element methods for non-selfadjoint elliptic problems: I. Error estimates. *Numer. Math.*, 72(4):501–522, 1996. doi:10.1007/s002110050179.

[55] S Reitzinger and J Schöberl. An algebraic multigrid method for finite element discretizations with edge elements. *Numer. Linear Algebr. with Appl.*, 9(3):223–238, 2002. doi:10.1002/nla.271.

[56] R N Rieben, D A White, B K Wallin, and J M Solberg. An arbitrary Lagrangian-Eulerian discretization of MHD on 3D unstructured grids. *J. Comput. Phys.*, 226(1):534–570, 2007. doi:10.1016/j.jcp.2007.04.031.

[57] Panayot S. Vassilevski. Sparse matrix element topology with application to amg and preconditioning. *Numer. Lin. Alg. Appl.*, 9:429–444, 2002.

[58] Panayot S Vassilevski. *Multilevel Block Factorization Preconditioners: Matrix-based Analysis and Algorithms for Solving Finite Element Equations*. Springer, New York, 2008. doi:10.1007/978-0-387-71564-3.

[59] Panayot S Vassilevski. Coarse Spaces by Algebraic Multigrid: Multigrid Convergence and Upscaling Error Estimates. *Adv. Adapt. Data Anal.*, 03(01n02):229–249, 2011. doi:10.1142/S1793536911000830.

[60] Panayot S Vassilevski and Umberto Villa. A Block-Diagonal Algebraic Multigrid Preconditioner for the Brinkman Problem. *SIAM J. Sci. Comput.*, 35(5):S3–S17, 2013. doi:10.1137/120882846.

[61] Panayot S. Vassilevski and Junping Wang. Multilevel iterative methods for mixed finite element discretizations of elliptic problems. *Numer. Math.*, 63(1):503–520, 1992. doi:10.1007/BF01385872.

[62] Jinchao Xu, Long Chen, and Ricardo H Nochetto. Optimal multilevel methods for $H(\text{grad})$, $H(\text{curl})$, and $H(\text{div})$ systems on graded and unstructured grids. In Ronald DeVore and Angela

Kunoth, editors, *Multiscale, Nonlinear Adapt. Approx.*, pages 599–659, Berlin, Heidelberg, 2009. Springer. doi:10.1007/978-3-642-03413-8_14.

[63] Stefano Zampini. PCBDDC: A Class of Robust Dual-Primal Methods in PETSc. *SIAM J. Sci. Comput.*, 38(5):S282–S306, 2016. doi:10.1137/15M1025785.

[64] Stefano Zampini and David E Keyes. On the Robustness and Prospects of Adaptive BDDC Methods for Finite Element Discretizations of Elliptic PDEs with High-Contrast Coefficients. In *Proc. Platf. Adv. Sci. Comput. Conf.*, New York, 2016. Association for Computing Machinery. doi:10.1145/2929908.2929919.

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551, USA.

Email address: kalchev1@llnl.gov

Department of Mathematics and Statistics, Portland State University, Portland, OR 97207, USA, and Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551, USA.

Email address: panayot@pdx.edu, vassilevski1@llnl.gov

Electrical & Systems Engineering, Washington University in St. Louis, St. Louis, MO 63130, USA.

Email address: uvilla@wustl.edu