HAL open science

Revisiting aggregation-based multigrid for edge elements

Artem Napov, Ronan Perrussel

▶ **To cite this version:**

Artem Napov, Ronan Perrussel. Revisiting aggregation-based multigrid for edge elements. Electronic Transactions on Numerical Analysis, 2019, 51, pp.118-134. 10.1553/etna_vol51s118 . hal-02388958

HAL Id: hal-02388958

https://hal.science/hal-02388958v1

Submitted on 5 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Electronic Transactions on Numerical Analysis.
Volume 51, pp. 118–134, 2019.
Copyright © 2019, Kent State University.
ISSN 1068–9613.
DOI: 10.1553/etna_vol51s118

ETNA
Kent State University and
Johann Radon Institute (RICAM)

REVISITING AGGREGATION-BASED MULTIGRID FOR EDGE ELEMENTS*

ARTEM NAPOV† AND RONAN PERRUSSEL‡

Abstract. We consider a modification of the Reitzinger-Schöberl algebraic multigrid method for the iterative solution of the curl-curl boundary value problem discretized with edge elements. The Reitzinger-Schöberl method is attractive for its low memory requirements and moderate cost per iteration, but the number of iterations typically tends to increase with the problem size. Here we propose several modifications to the method that aim at curing the size-dependent convergence behavior without significantly affecting the attractive features of the original method. The comparison with an auxiliary space preconditioner, a state-of-the-art solver for the considered problems, further indicates that both methods typically require a comparable amount of work to solve a given discretized problem but that the proposed approach requires less memory.

Key words. algebraic multigrid, edge elements, preconditioning, aggregation

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

1. Introduction.

We consider algebraic multigrid methods for the iterative solution of large sparse $n \times n$ symmetric positive definite (SPD) linear systems

$$A\mathbf{u} = \mathbf{b} \tag{1.1}$$

obtained from the discretization of the boundary value problem

$$\left\{ \begin{array}{rcll} \mathbf{curl}\left(\mu^{-1}\,\mathbf{curl}\,\mathbf{E}\right) + \beta\,\mathbf{E} & = & \mathbf{f} & \text{in } \Omega, \\ \mathbf{E} \times \mathbf{n} & = & \mathbf{g}_0 & \text{on } \Gamma_D \subset \partial\Omega, \\ (\mu^{-1}\,\mathbf{curl}\,\mathbf{E}) \times \mathbf{n} & = & \mathbf{g}_1 & \text{on } \Gamma_N \;=\; \partial\Omega \backslash \Gamma_D, \end{array} \right. \tag{1.2}$$

with edge elements. Such a boundary value problem arises, for instance, in the computation of a slowly-varying electromagnetic field with implicit integration in time. Then, $\mathbf{E}$ represents an electric field in a bounded simply connected region Ω of $\mathbb{R}^2$ or $\mathbb{R}^3$, $\mu > 0$ is the magnetic permeability, $\beta > 0$ is the ratio of the electric conductivity and the integration time step, and $\mathbf{n}$ represents the unit outward normal vector to the boundary surface $\partial\Omega = \Gamma_D \cup \Gamma_N$ [3]. By edge elements we mean the lowest-order elements of the first family proposed by Nédélec on simplex and brick meshes [18]; in both cases the degrees of freedom are associated with the individual edges of the mesh.

An important feature of the resulting system matrix is a large near null space. Its presence can already be deduced from the original boundary value problem (1.2), in which the leftmost $\mathbf{curl}$-$\mathbf{curl}$ term vanishes for any function of the form $\mathbf{grad}\,\phi$; that is, such functions belong to the null space of the $\mathbf{curl}$ operator. Actually, since Ω is a simply connected domain, the space of all gradients on Ω exactly corresponds to the null space of the $\mathbf{curl}$ operator and therefore forms the near null space of the continuous boundary value operator in (1.2). In a similar way, the near null space of the system matrix A corresponds to the null space of the discrete $\mathbf{curl}$ operator, which in turn is formed by all the vectors of the form $G\mathbf{v}$, where G is the discrete gradient operator. Moreover, assuming a proper normalization, the discrete gradient G is then given by the nodes-to-edges incidence matrix of the underlying mesh [6], and therefore the near null space of A is actually known.

*Received January 16, 2018. Accepted January 15, 2019. Published online on May 10, 2019. Recommended by Stefan Vandewalle.

†Université Libre de Bruxelles, Service de Métrologie Nucléaire (C.P. 165-84), 50 Av. F.D. Roosevelt, B-1050 Brussels, Belgium (anapov@ulb.ac.be).

‡LAPLACE, Université de Toulouse, CNRS, INPT, UPS, Toulouse, France (perrussel@laplace.univ-tlse.fr).

Now, the presence of a large near null space makes the iterative solution of the resulting linear systems challenging. This is because the presence of a near null space undermines the convergence of standard single-level iterative solvers. Regarding multilevel methods, such a near null space is also a difficulty for the *standard* multigrid and multilevel methods [8, 27]. Indeed, these latter methods are designed to only cope with a small near null space, typically composed of one or few given vectors.

Hence, we resort to *special* multigrid methods which are specifically designed for the discretizations of (1.2). These methods follow the main multigrid ideas: a single-level iterative scheme (*smoother*) is combined with a solution of a smaller linear system (*coarse grid correction*), and, if the smaller system is big enough, it is solved only approximately with a few recursive iterations of the multigrid method, therefore requiring a hierarchy of more than two systems. However, the special multigrid methods have one of the two following distinctive features: either the smoother and the coarse grid correction are chosen to properly handle the large near null space and therefore differ from standard multigrid components, or the preconditioning of the original system (1.1) amounts to precondition some extended and/or auxiliary systems with a standard (i.e., not special) multigrid method. The methods implementing the first feature include those from [1, 9], which consider the discrete boundary value problem (1.2) on simplex meshes that are obtained by a successive refinement of a given mesh, that is, in a *geometric* setting. In such a setting the coarse grid correction is naturally defined by the resulting mesh hierarchy, whereas each method employs its own special smoother. When the mesh hierarchy is not available, it should be built by the method itself during the setup (coarsening) stage; this is done by *algebraic* multigrid (AMG) methods. The special AMG methods then typically use one of the smoothers from [1, 9], whereas the grid hierarchy is deduced from the one built with a standard AMG method for an auxiliary "nodal" matrix. This approach was proposed in [26] by Reitzinger and Schöberl, further improvements have been suggested in [4, 5, 11, 24, 26]. Most of the resulting methods require comparatively little memory and comparatively little work per iteration, but their convergence often deteriorates with the problem size, albeit the deterioration is sometimes slow. Some of the methods do actually achieve a level-independent convergence but at the price of a significant increase in complexity and memory usage.

The special multigrid methods that use a second distinctive feature, namely the preconditioning of some extended/auxiliary systems with standard multigrid, include [2, 10, 12, 13, 14]. Amongst these, *auxiliary space preconditioners*, some of which are studied in [13, 14] whereas others are analyzed in [10], are deemed state-of-the-art solvers for the considered problems. For these problems, they exhibit a level-independent convergence, but due to the use of large auxiliary matrices, they typically require more memory and perform more work per iteration than most of the multigrid methods based on the first distinctive feature.

In this work, we revisit the aggregation-based method [26] by Reitzinger and Schöberl. This is a special multigrid method with the first distinctive feature: it handles the large near null space by combining a special hybrid smoother from [1, 9] with a properly built aggregation-based grid hierarchy. The aggregation procedure is designed to yield a prolongation operator that satisfies an important commutativity property, which is also known to hold in the geometric setting of [1, 9].

We modify the Reitzinger-Schöberl method so as to achieve a fast and size-independent convergence, while managing to preserve the attractive memory usage and cost per iteration of the original method. For this, we follow the ideas from [16, 20, 23], where a proper choice of multigrid ingredients enables fast and size-independent convergence for the aggregation-based standard multigrid. More specifically, we use a proper cycling strategy for the multigrid

recursion, consider a suitable auxiliary matrix for the construction of "nodal" aggregates and apply a state-of-the-art aggregation scheme to this auxiliary matrix.

The resulting method is then compared with an auxiliary space preconditioner based on [14]. As said, variants of auxiliary space preconditioners are considered nowadays as reference solvers for the problems arising from the edge element discretization of (1.2) and have provably level-independent convergence for these applications. The outcome of the comparison is that both the proposed method and an auxiliary space preconditioner require approximately the same amount of work to solve the system (1.1), although the auxiliary space preconditioner is typically slightly better in this regard. However, the proposed method always needs substantially less memory.

The remainder of this paper is structured as follows. In Section 2 we review the Reitzinger-Schöberl multigrid method. In Section 3 we summarize the modifications of the method needed for a better convergence and robustness. The performance of the resulting method is then assessed on a set of various problems in Section 4, including problems on unstructured meshes, with mesh refinement, and with discontinuities in the coefficients μ^{-1} and β. The method is compared with an auxiliary space preconditioner based on [14] in Section 5 and the conclusions are stated in Section 6.

Note that, for the reader's convenience, in Section 3 the proposed modifications to the Reitzinger-Schöberl multigrid method are only outlined. The motivation behind these modifications can, however, be found in the associated report [17], which represents an earlier and more detailed version of this work. Another significant difference with respect to the report [17] is the use of a different version of the `gmsh` mesh generator [7] for the numerical experiments (version `3.0.6` here versus `2.4.2` in [17]), which produces different discretization meshes. The fact that this latter change has little impact on the reported results further highlights the robustness of the considered methods.

2. Reitzinger-Schöberl multigrid method. In this section we review the basic multigrid components as implemented in the Reitzinger-Schöberl multigrid method. In particular, as any multigrid method is obtained by a recursive application of a two-grid scheme, we review the corresponding Reitzinger-Schöberl two-grid scheme first.

2.1. Two-grid scheme. A two-grid scheme is a combination of *smoothing iterations* and a *coarse grid correction*. A smoothing iteration is typically one iteration of a single-level iterative method; for instance, Gauss-Seidel smoothers are commonly employed within the standard multigrid methods. The coarse grid correction amounts to solve a smaller problem on a coarser grid. A proper interplay of the smoothing iterations and the coarse grid correction is the key to the efficiency of the two-grid scheme.

A two-grid iteration matrix, which combines a pre-smoothing iteration, a coarse grid correction, and a post-smoothing iteration, is given by

$$I - B_{\mathrm{TG}}A \;=\; (I - M^T A)(I - PA_c^{-1}P^T A)(I - MA), \tag{2.1}$$

where M is the preconditioner for the smoothing iteration, P is the prolongation matrix, and $A_c = P^T AP$ is the coarse grid matrix. Regarding more specifically the coarse grid correction step, it proceeds by first transferring the residual vector to the coarse grid by applying the transpose of the prolongation matrix P^T to it, then solving the coarse residual system with the system matrix A_c, and finally transferring the resulting correction from the coarse grid by applying the prolongation matrix P to it. Note that, for a given A, the coarse grid correction step is completely determined by the prolongation matrix P.

Now, we intend to use multigrid methods as preconditioners for the Conjugate Gradient (CG) iteration. Such a use is typically advised for algebraic multigrid methods as it increases

their robustness. Regarding the two-grid scheme, the corresponding preconditioner B_{TG} can be determined from (2.1) and is given by

$$B_{\text{TG}} = \overline{M} + (I - M^T A) P A_c^{-1} P^T (I - AM), \tag{2.2}$$

where

$$\overline{M} = M + M^T - M^T AM.$$

Of course, in such a setting one needs to make sure that the two-grid preconditioner is SPD. However, it is then enough to verify that the matrix $\overline{M}$ is SPD or, equivalently, that the smoothing iteration based on M is converging [28, Chapter 3].

Now, the two-grid Reitzinger-Schöberl method fits into the above description but requires a special smoother and coarse grid correction to cope with the large near null space of the discrete boundary value problem (1.2). This latter space is spanned by the range of the discrete gradient operator G, which is typically available. In particular, if the degrees of freedom are properly normalized, as assumed in this work, then the discrete gradient G is given by the nodes-to-edges incidence matrix of the underlying mesh, that is, each row of G has at most two nonzero entries 1 and -1, corresponding respectively to the ending node and the starting node of the associated edge in the mesh.

Regarding the special smoother, in what follows we use the hybrid Gauss-Seidel smoother from [9]. It corresponds to a combination of one Gauss-Seidel iteration applied to the system projected into the range of the gradient operator and of one Gauss-Seidel iteration applied to the original system. The iteration matrix for this smoother is given by

$$I - MA = (I - L^{-1}A)(I - G\,(L^{(n)})^{-1} G^T A), \tag{2.3}$$

where $L^{(n)}$ and L represent, respectively, the lower part of $G^T AG$ and A. Moreover (as shown in [17]), the two-grid preconditioner based on such a smoother is SPD and therefore can precondition the CG iteration.

Regarding the special coarse grid correction, as said, it is determined by the prolongation matrix, and this matrix is of aggregation type in the Reitzinger-Schöberl method. A prolongation of aggregation type requires the unknowns of the original system to be grouped into disjoint sets (or aggregates) with each set corresponding to an unknown of the coarse system. The application of the prolongation matrix then amounts to assign to all the unknowns in the k-th aggregate the weighted value of the k-th coarse unknown; the non-aggregated unknowns are set to 0.

The construction of the prolongation matrix within the Reitzinger-Schöberl method is done in two steps. At first, the preliminary "nodal" aggregates are built from the auxiliary unknowns that are associated with the nodes of the discretization mesh. Such "nodal" unknowns typically correspond to the columns of the gradient operator as available on the fine mesh. This means, in particular, that each original (or "edge") unknown j correspond to two "nodal" unknowns j_1 and j_2, the first being the starting point and the second the ending point of the edge; this relation is denoted below as $j = (j_1, j_2)$. This first step is commonly performed by applying a classical algebraic aggregation method to an auxiliary "nodal" matrix, for example, the matrix resulting from the discretization of the div-grad operator with Lagrange linear finite elements on the same mesh as the considered discrete problem.

The second step amounts to building the prolongation matrix out of the auxiliary "nodal" aggregates, which in what follows we denote K_k, $k = 1, \ldots, n_c^{(n)}$. Since the prolongation is of aggregation type, we first explain how the corresponding "edge" aggregates are built, then specify the prolongation weights, and finally give the prolongation matrix itself. Every

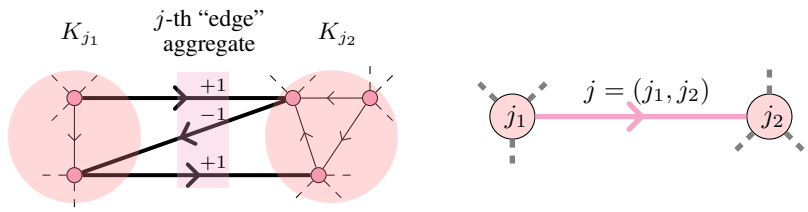

FIGURE 2.1. *Example of the "nodal" and "edge" aggregates on the fine grid with the corresponding prolongation weights (left) and the associated coarse grid mesh (right).*

nonempty "edge" aggregate corresponds to two "nodal" aggregates K_{j_1} and K_{j_2} having at least one connecting edge; the aggregate gathers all the unknowns associated to edges that have one vertex in K_{j_1} and another in K_{j_2}; see Figure 2.1 for an example. Such an "edge" aggregate defines a coarse unknown j which corresponds to a coarse edge that relates the coarse nodes j_1 and j_2. That is, fixing the orientation, we have a coarse nodes-to-edge relation $j = (j_1, j_2)$, which further specifies the coarse gradient G_c. As of the prolongation weights, they are set to 1 or -1 depending on whether the edge unknown from the aggregate has the same orientation as its coarse representation. Put together, this defines the Reitzinger-Schöberl prolongation matrix

$$(P)_{ij} = \begin{cases} 1 & \text{if } i_1 \in K_{j_1} \text{ and } i_2 \in K_{j_2}, \\ -1 & \text{if } i_1 \in K_{j_2} \text{ and } i_2 \in K_{j_1}, \\ 0 & \text{otherwise,} \end{cases}$$

where $i = (i_1, i_2)$ is a fine unknown and $j = (j_1, j_2)$ is a coarse unknown. Note that the above prolongation matrix satisfies an important commutativity relation [26], which is also known to hold in the geometric multigrid setting.

2.2. Multigrid. A multigrid method is obtained from a two-grid scheme when the coarse problem is solved approximately with one or few recursive applications of the multigrid method on a coarse level. More specifically, this means that the exact solution of the coarse system, as represented by the application of A_c^{-1} in the expressions of the two-grid iteration matrix (2.1) and the two-grid preconditioner (2.2), is replaced with an approximate solution obtained with one or few steps of the multigrid preconditioner on the coarse level. Of course, to define such a preconditioner, an even smaller coarse system is required and so on, resulting in a hierarchy of increasingly smaller systems; the coarsest system is chosen small enough to be solved exactly at a negligible cost.

Practical multigrid methods are designed to be optimal solvers in the sense that the cost of the solution of the linear system (1.1) is proportional to the number $\text{nnz}(A)$ of nonzero entries of the system matrix. Optimality is achieved when, on the one hand, the cost per preconditioner application is proportional to $\text{nnz}(A)$, and, on the other hand, the number of iterations needed for convergence is bounded independently of the problem size. The former condition is satisfied if the complexity parameter $\mathcal{C}_{\mathcal{W}}$ introduced below is also bounded independently of the problem size.

The Reitzinger-Schöberl method is an algebraic multigrid (AMG) method, which means that the hierarchy of coarse systems is built based solely on the system matrix $A_1 = A$ and the corresponding discrete gradient operator $G_1 = G$. Note that such a hierarchy is completely determined by specifying at every level the procedure for the construction of the "nodal" aggregates. Indeed, as follows from the discussion in Section 2.1, once the "nodal" aggregates are built at a given level ℓ, $\ell = 1, \ldots, L-1$, the edge prolongation P_ℓ and the coarse gradient

operator $G_{\ell+1}$ can be determined. This further yields the system matrix on the next coarse level via $A_{\ell+1} = P_\ell^T A_\ell P_\ell$ and the same procedure can therefore be repeated at this level.

3. Reitzinger-Schöberl revisited. In this section we summarize the considered modifications of the Reitzinger-Schöberl multigrid method: the use of K-cycle multigrid and a new algebraic procedure for the construction of "nodal" aggregates. The motivation behind these modifications is given in [17].

The first modification—the use of K-cycle multigrid—aims at curing the deterioration of convergence that is typically observed for the Reitzinger-Schöberl multigrid method with the refinement of the discretization mesh [26]. The source of such a deterioration lies in the fact that the approximate multigrid solution of the coarse system, as carried out during the coarse grid correction step, is not accurate enough. Indeed (as observed in [15, 17]) the convergence of the Reitzinger-Schöberl *two*-grid method, in which the coarse system is solved exactly, does not suffer from such a deterioration.

Now, the accuracy of the approximate solution of the coarse system essentially depends on the multigrid cycle, that is, on the number and nature of the multigrid iterations recursively used to approximately solve the coarse system. The original Reitzinger-Schöberl multigrid is used with the V-cycle that is obtained when at every but the last level the solution of the coarse system is replaced with *one* application of the multigrid preconditioner. Compared to the other multigrid cycles, a V-cycle always needs less work per application. On the negative side, an optimal convergence is not always easy to ensure with the V-cycle multigrid. In particular, it is known that for optimal convergence, the standard multigrid methods require a prolongation operator which is at least linear [27], this latter condition being violated by aggregation-based standard multigrid methods.

Aggregation-based standard multigrid methods are typically used in combination with the K-cycle [16, 20]. K-cycle multigrid is obtained when at every but the last level the solution of the coarse system is replaced with *two* applications of the multigrid preconditioner *accelerated* by the Flexible Conjugate Gradient method [19]; the flexible variant is required since the resulting preconditioner is slightly non-linear. The work for one application of the multigrid preconditioner is then typically higher than for a V-cycle, but the increase is typically moderate if the size of the coarse system decreases by a factor of 4 or more from one level to the next.

Regarding the convergence of the resulting K-cycle Reitzinger-Schöberl multigrid, it can typically be characterized as independent of the problem size. This is also illustrated in Section 4 below, where we systematically report the number of iterations for three different sizes of each problem. Further, the overall cost to solution is also typically lower for the K-cycle variant of the Reitzinger-Schöberl multigrid compared to the V-cycle variant [17].

We now briefly comment on a parallel perspective of the K-cycle multigrid. It is commonly believed that a scalable multigrid solver on a massively parallel architecture can only be obtained with the V-cycle multigrid. This is perhaps true if one considers a straightforward parallelization of the sequential algorithm. However, it is demonstrated in [22] that a properly redesigned K-cycle multigrid can also be scalable on modern massively parallel machines. As the ideas behind the approach in [22] are not specific to Poisson-like problems, they can also be applied for the parallelization of a Reitzinger-Schöberl K-cycle multigrid method.

The second modification—a new procedure for the construction of the "nodal" aggregates—aims at making the method more robust. Note that such a robustness is considered here in a rather challenging context of an algebraic multigrid (AMG) method for the discretizations of (1.2), for which only the original system (1.1) and the associated discrete gradient operator G are available.

Now, as mentioned above, a construction procedure of the "nodal" aggregates completely determines the hierarchy of coarse systems in the Reitzinger-Schöberl multigrid method. Here,

the auxiliary "nodal" aggregates are determined at level ℓ, $\ell = 1, \ldots, L-1$, by applying a standard aggregation procedure to the auxiliary "nodal" matrix $G_\ell^T A_\ell G_\ell$. Regarding the aggregation procedure, we consider the multiple pairwise aggregation with quality control for symmetric system matrices [16] as implemented in the `3.2.3-aca` release of the `AGMG` software package [21]. This aggregation algorithm proceeds in several passes: during the first pass, the "nodal" unknowns are grouped into aggregates of size one or two, and during the subsequent passes, at each pass some pairs of aggregates are merged into single aggregates. The resulting aggregates are therefore of size at most 4 after 2 passes, at most 8 after 3 passes, etc. The choice of a tentative pair of aggregates and the decision to merge them are based on a measure of the aggregate's quality, and only aggregates with the quality measure below $\overline{\kappa}$ are accepted. As here we target mostly three-dimensional problems and aim at forming large enough aggregates to favor low complexities, we use the considered aggregation scheme with 3 passes and with the threshold value $\overline{\kappa} = 20.0$.

Note, however, that the resulting aggregates are only accepted if their average aggregate size for a given level is larger than 4; in the infrequent case where this condition is not satisfied, the multiple pairwise aggregation is run again at this level but this time with the auxiliary matrix $G_\ell^T G_\ell$. Explanations and numerical experiments are provided in [17] to justify this latter choice; this particular modification is denoted there $(G^T AG)_*$ to distinguish it from the use of $G^T AG$ alone.

4. Numerical results. In this section we report on numerical experiments with the variant of the Reitzinger-Schöberl method described in Section 3. Our experiments cover the discretizations of (1.2) on structured and unstructured meshes in two and three dimensions, separately considering the effect of local mesh refinement and a discontinuity in either the magnetic permeability coefficient μ or in the coefficient β. For all the considered problems we set $\beta = 10^{-2}$ and $\mu^{-1} = 1$ unless stated otherwise. The number of iterations is that of the preconditioned Flexible Conjugate Gradient method needed to reduce the relative Euclidean norm of the residual by the factor of 10^6; the initial residual is chosen as a random vector generated starting from a fixed seed. The unstructured meshes are generated with the `3.0.6` release of the `gmsh` package [7], and the corresponding system matrices are assembled with the `4.1` release of the `getfem++` software [25].

Now, to correctly interpret the reported results one needs not only the number of iterations but also the work per iteration. The work estimate is provided here with the complexity parameter $\mathcal{C}_{\mathcal{W}}$, which corresponds to the number of floating-point multiplications needed for one application of the preconditioner under consideration normalized by the number of nonzero entries of the preconditioned system matrix A. For the Reitzinger-Schöberl multigrid method this parameter is approximately given by

$$\mathcal{C}_{\mathcal{W}} = \sum_{\ell=1}^{L-1} 2^{\ell-1} \frac{5\,\mathrm{nnz}(A_\ell) + \mathrm{nnz}(A_\ell^{(n)})}{\mathrm{nnz}(A)} + 2^{L-1} \frac{\mathrm{nnz}(A_L)}{\mathrm{nnz}(A)}, \tag{4.1}$$

where $A_1 = A$, A_ℓ is the coarse system matrix at level ℓ, $\ell = 2, \ldots, L$, and $A_\ell^{(n)} = G_\ell^T A_\ell G_\ell$ is the auxiliary "nodal" matrix at level ℓ, $\ell = 1, \ldots, L-1$. The derivation of the above expression is given in the appendix. Note that in order to keep the complexity parameter $\mathcal{C}_{\mathcal{W}}$ bounded, one needs to ensure that $\mathrm{nnz}(A_\ell)$ and $\mathrm{nnz}(A_\ell^{(n)})$ decrease by a factor larger than 2 from one level to the next; in practice, the decrease by a factor 4 in the dimension of the coarse matrix A_ℓ is often enough to ensure a proper complexity.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.1
Convergence, complexity, and memory parameters for the considered method with various boundary value problems discretized on structured meshes. The parameter h indicates the mesh size. The parameters $\mathcal{C}_{\mathcal{W}}$ and $\mathcal{C}_{\mathcal{M}}$ are, respectively, the estimate (4.1) of the work per one preconditioner application and the estimate (4.2) of the memory usage, both normalized by nnz(A).

	MODB_2D_D			MODB_3D_D			MODS_2D_N			MODS_3D_N		
h^{-1}	400	800	1600	20	40	80	400	800	1600	20	40	80
n (10^5)	3.2	12.8	51.2	0.2	1.7	14.4	4.8	19.1	76.7	0.5	4.3	35.1
nnz/n	7.0	7.0	7.0	30.2	31.7	32.2	5.0	5.0	5.0	15.7	16.1	16.2
#it	4	4	4	16	19	21	52	59	60	22	24	26
$\mathcal{C}_{\mathcal{W}}$	7.4	7.5	7.5	8.2	9.5	10.4	7.2	7.2	7.2	6.5	6.7	6.7
$\mathcal{C}_{\mathcal{M}}$	1.9	1.9	1.9	1.6	1.7	1.8	1.7	1.6	1.6	1.3	1.3	1.3

Finally we also report the memory consumption via the parameter

$$\mathcal{C}_{\mathcal{M}} = \sum_{\ell=1}^{L-1} \frac{\text{nnz}(A_\ell) + \text{nnz}(A_\ell^{(n)})}{\text{nnz}(A)} + \frac{\text{nnz}(A_L)}{\text{nnz}(A)}, \tag{4.2}$$

which represents the memory used for the storage of all the floating-point data normalized by the memory required for the system matrix A. This interpretation is stated under the assumption, made in this work, that the matrices $A_\ell^{(n)}$, $\ell = 1, \dots, L-1$, are stored explicitly in memory; we also neglect the contribution of G_ℓ and P_ℓ, $\ell = 1, \dots, L-1$, since the nonzero entries of these matrices are given by ± 1.

4.1. Structured mesh. We begin by examining the performance of the considered method when applied to the discretizations of (1.2) on structured meshes. The considered problems are defined on a unit square (2D) or a unit cube (3D), with Dirichlet ($\Gamma_D = \partial\Omega$) or natural ($\Gamma_N = \partial\Omega$) boundary conditions (D or N, respectively) and discretized on brick (MODB) or simplex (MODS) meshes.

The results for these problems are reported in Table 4.1. The main observation is that both the iteration count and the complexity parameter $\mathcal{C}_{\mathcal{W}}$ seem to be bounded independently of the problem size, suggesting that the considered Reitzinger-Schöberl method is indeed optimal for the considered problems. This observation remains valid for the other numerical experiments in this work, for both brick and simplex meshes. As a side comment, we note that the parameters $\mathcal{C}_{\mathcal{W}}$ and $\mathcal{C}_{\mathcal{M}}$ are slightly higher for brick meshes, meaning that more work per iteration and memory is required per nonzero entry of A. However, this comes together with lower iteration counts.

4.2. Unstructured mesh. We now consider the problems discretized on simplex meshes. The considered problems are both two- and three-dimensional (2D and 3D), and in every case we consider one boundary value problem (1.2) discretized on a structured (MODS) and an unstructured (SQR or CUB) mesh. More precisely, problems MODS_2D_N and SQR_2D_N are defined on a unit square with natural boundary conditions ($\Gamma_N = \partial\Omega$), whereas MODS_3D_D and CUB_3D_D are defined on a unit cube with Dirichlet ($\Gamma_D = \partial\Omega$) boundary conditions.

The results for the considered problems are reported in Table 4.2. Comparing the respective columns for the two- and the three-dimensional problems, we see that going from a structured to the corresponding unstructured mesh slightly increases the complexity and memory parameters and slightly decreases the number of iterations. We therefore conclude that unstructured meshes do not represent a difficulty for the considered method.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.2
Convergence, complexity, and memory parameters for the considered method with various boundary value problems discretized on simplex meshes. The parameter $h_{\max}$ indicates the maximal mesh size for the unstructured meshes and the actual mesh size for the structured ones.

	MODS_2D_N			SQR_2D_N			MODS_3D_D			CUB_3D_D		
$h_{\max}^{-1}$	400	800	1600	20	40	80	40	80	160	20	40	80
n (10^5)	4.8	19.1	76.7	3.6	14.3	58.1	0.4	4.0	34.0	0.3	3.4	21.3
nnz/n	5.0	5.0	5.0	5.0	5.0	5.0	15.6	16.0	16.2	15.7	16.2	16.4
#it	52	59	60	37	39	42	23	27	29	22	24	26
$\mathcal{C}_{\mathcal{W}}$	7.2	7.2	7.2	8.0	8.1	8.1	6.8	6.8	6.8	7.9	8.0	8.0
$\mathcal{C}_{\mathcal{M}}$	1.7	1.6	1.6	1.7	1.7	1.8	1.3	1.3	1.3	1.4	1.4	1.4

TABLE 4.3
Convergence, complexity, and memory parameters for the considered method with various boundary value problems discretized on simplex meshes. The parameter $h_{\max}$ indicates the maximal mesh size.

	SQR_2D_N			SQR_R_2D_N			CUB_3D_D			CUB_R_3D_D		
$h_{\max}^{-1}$	20	40	80	20	40	80	20	40	80	20	40	80
n (10^5)	3.6	14.3	58.1	11.4	45.8	183.1	0.3	3.4	21.3	1.1	6.1	45.2
nnz/n	5.0	5.0	5.0	5.0	5.0	5.0	15.7	16.2	16.4	16.0	16.2	16.3
#it	37	39	42	45	49	54	22	24	26	25	26	27
$\mathcal{C}_{\mathcal{W}}$	8.0	8.1	8.1	8.1	8.1	8.1	7.9	8.0	8.0	7.9	7.9	8.0
$\mathcal{C}_{\mathcal{M}}$	1.7	1.7	1.8	1.8	1.8	1.8	1.4	1.4	1.4	1.4	1.4	1.4

4.3. Mesh with local refinement. Here we examine the impact of a local mesh refinement on the performance parameters. For this, we modify the problems SQR_2D_N and CUB_3D_D as introduced in the previous section by refining their mesh, respectively, by a factor of 20 near the center of the square domain and by a factor of 10 near the center of the cube domain; the new problems are referred to as SQR_R_3D_N and CUB_R_3D_D.

The results for these problems are reported in Table 4.3. We note that the local mesh refinement has little impact on the complexity of the considered algorithm. On the other hand, it typically increases the iteration count. This is probably due to a higher fraction of poor-shaped simplexes in the meshes with local refinement compared to the meshes without it.

4.4. Discontinuity. Finally we consider the effect of discontinuities in the magnetic permeability coefficient μ on the complexity and convergence parameters. For this, we report these parameters for the problems which only differ by the presence of discontinuities in μ. Regarding the problems on structured brick meshes, we consider the already introduced model problem MODB_3D_D on a cube with Dirichlet boundary conditions and compare it with the same problem MODB_J_3D_D in which the coefficient μ^{-1} varies from 1 to 10^{-3} through the $x = 0.5$ plane. For problems on unstructured simplex meshes, we reconsider the CUB_3D_N problem on a unit cube with natural boundary conditions and compare it to the same problem with μ^{-1} given by

$$\begin{cases} 1 & \text{for } x > 0.5 \text{ and } y, z < 0.5, \\ 20^{-1} & \text{for } x, z > 0.5 \text{ and } y < 0.5, \\ 20^{-2} & \text{for } x, y > 0.5 \text{ and } z < 0.5, \\ 20^{-3} & \text{for } x, y, z > 0.5, \end{cases} \qquad \begin{cases} 20^{-4} & \text{for } x, y, z < 0.5, \\ 20^{-5} & \text{for } x, y < 0.5 \text{ and } z > 0.5, \\ 20^{-6} & \text{for } x < 0.5 \text{ and } y, z > 0.5, \\ 20^{-7} & \text{for } x, z < 0.5 \text{ and } y > 0.5. \end{cases}$$

The results for these problems are reported in Table 4.4. One may see that there is practically no difference between the problems with or without discontinuity, be it in the case of structured brick meshes or in the case of unstructured simplex ones. Moreover, the same

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.4
Convergence, complexity, and memory parameters for the considered method with various three-dimensional boundary value problems with discontinuities. The parameter $h_{\max}$ indicates the maximal mesh size for the unstructured meshes and the actual mesh size for the structured ones.

	MODB_3D_D			MODB_J_3D_D			CUB_3D_N			CUB_J_3D_N		
$h_{\max}^{-1}$	20	40	80	20	40	80	20	40	80	20	40	80
n (10^5)	0.2	1.7	14.4	0.2	1.7	14.4	0.4	3.8	22.8	0.4	3.8	22.8
nnz/n	30.2	31.7	32.3	30.2	31.7	32.3	15.3	15.8	16.1	15.3	15.8	16.1
#it	16	19	21	16	19	21	21	24	25	21	22	23
$\mathcal{C}_{\mathcal{W}}$	8.2	9.5	10.4	8.2	9.5	10.4	7.4	7.6	7.8	7.4	7.6	7.8
$\mathcal{C}_{\mathcal{M}}$	1.6	1.7	1.8	1.6	1.7	1.8	1.4	1.4	1.4	1.4	1.4	1.4

TABLE 4.5
Convergence, complexity, and memory parameters for problems with discontinuities in coefficients μ^{-1} and β. The parameter $h_{\max}$ indicates the maximal mesh size.

	CUB_3D_N			CUB_JA_3D_N			CUB_JB_3D_N			CUB_JC_3D_N		
$h_{\max}^{-1}$	20	40	80	20	40	80	20	40	80	20	40	80
#it	21	24	25	24	26	27	26	27	29	27	30	32
$\mathcal{C}_{\mathcal{W}}$	7.4	7.6	7.8	7.4	7.7	7.8	7.4	7.7	7.8	7.4	7.6	7.8
$\mathcal{C}_{\mathcal{M}}$	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4

behavior is observed for other test problems with discontinuities, including two-dimensional problems. Whereas these observations are to be expected for the complexity and memory parameters since the contribution of the discrete curl-curl term to A vanishes within $G^T AG$ and so is the dependence on the μ coefficient, the fact that the number of iterations is independent of the discontinuity seems rather unexpected.

Now, in the above problems only μ^{-1} is considered discontinuous. Partially, this is because the contribution of the β-multiplied term (or mass term) in the discretization of (1.2) is typically small, whereas in some applications this term comes from the regularization of a singular version of (1.2) corresponding to $\beta = 0$ and therefore remains continuous. However, since the nodal aggregation is only dependent on this term and not on the discrete curl-curl term, it is still instructive to consider the situation where β is discontinuous. This is done in Table 4.5, in which the original CUB_3D_N problem with continuous coefficients is compared with the problems CUB_JA_3D_N, CUB_JB_3D_N and CUB_JC_3D_N. The second (JA) and the third (JB) problems are the same as the first one except that μ^{-1} varies from 1 to 10^{-3} through the plane $x = 0.5$, whereas β varies from 10^{-2} to 10^{-5} through the plane $x = 0.5$ (for the CUB_JA_3D_N problem) or through the plane $y = 0.5$ (for the CUB_JB_3D_N problem). Note that since in the second case the discontinuities in μ^{-1} and β are located one perpendicular to another, the CUB_JB_3D_N problem is both artificial and quite challenging. The fourth problem (JC) correspond to a continuous μ^{-1} ($\mu^{-1} = 1$) but discontinuous β, the latter jumping from 1 to 10^{-8} when crossing the plane $x = 0.5$.

Regarding the results, we note that, although the aggregates produced for the three problems are now different, the results remain comparable. This essentially indicates that the location of the nodal aggregates with respect to the discontinuity has a limited impact on the overall convergence of the method.

5. Comparison with an auxiliary space preconditioner.

5.1. Auxiliary space preconditioners. Auxiliary space preconditioners [10, 13, 14] are considered today as reference methods for the solution of discrete boundary value problems of the form (1.2). They amount to applying one step of a standard multigrid preconditioner to

several auxiliary linear systems. This ability to rely on well-established standard multigrid methods is one of the attractive points of auxiliary space preconditioners.

An auxiliary space preconditioner is only partially algebraic in the aforementioned sense, since, in addition to A and G, it also requires the user to provide an interpolation matrix Π, which maps the vector degrees of freedom at the nodes of the discretization mesh to the edge degrees of freedom. For a three-dimensional problem the interpolation matrix can typically be represented as

$$\Pi \;=\; \begin{bmatrix}\Pi_x & \Pi_y & \Pi_z\end{bmatrix},$$

where Π_x has the same sparsity structure as G with the nonzero entries in the row i (at most two) being given by $(G\mathbf{x})_i/2$ with $\mathbf{x}$ being the vector of x-coordinates of the nodes of the mesh; the matrices Π_y and Π_z are determined similarly. As a result, the matrices Π, Π_x, Π_y, and Π_z can be deduced from the discrete gradient G and the coordinates of the mesh nodes.

Following the ideas from [13, 14] we consider more specifically a multiplicative variant of an auxiliary space preconditioner with Galerkin projections on subspaces and with additive scalar decomposition. Multiplicative variant means that the corrections are performed successively, not simultaneously; the corresponding iteration matrix is then the product of the iteration matrices of the individual corrections. The additive scalar decomposition means that some auxiliary matrices are induced by the scalar components Π_x, Π_y (and Π_z in three dimensions) and the corresponding corrections are performed simultaneously, not successively. The iteration matrix of the corresponding preconditioner B_{asp} in three dimensions is given by

$$\begin{aligned} I - B_{\text{asp}}A = {} & (I - GB^{(n)}G^TA)(I - L^{-T}A) \\ & * \left(I - \left(\sum_{i\in\{x,y,z\}} \Pi_i B_{\Pi_i} \Pi_i^T\right) A\right)(I - L^{-1}A)(I - GB^{(n)}G^TA), \end{aligned} \tag{5.1}$$

where $B^{(n)}$ and B_{Π_i}, $i \in \{x, y, z\}$, are the standard multigrid preconditioners for the matrices $A^{(n)} = G^TAG$ and $A_{\Pi_i} = \Pi_i^T A\Pi_i$, $i \in \{x, y, z\}$. Note that although the first two factors in (5.1) look similar to the iteration matrix (2.3) of the hybrid smoother, here the multigrid preconditioner $B^{(n)}$ cannot be replaced by a simple Gauss-Seidel relaxation $(L^{(n)})^{-1}$, with $L^{(n)} = \text{tril}(G^TAG)$; such a replacement typically ruins the attractive convergence properties.

To make the comparison of the auxiliary space preconditioner defined by (5.1) and the Reitzinger-Schöberl method considered here more complete, we introduce below the complexity and memory parameters which play a similar role as the parameters $\mathcal{C}_{\mathcal{W}}$ and $\mathcal{C}_{\mathcal{M}}$ defined by (4.1) and (4.2), respectively. Regarding the complexity parameter, it reflects the work per nonzero entry of A needed for one application of the auxiliary space preconditioner and is approximately given by

$$\mathcal{C}_{\mathcal{W}} \;=\; \frac{5\,\text{nnz}(A) + 6\,\text{nnz}(A^{(n)})\,\widetilde{\mathcal{C}}_{\mathcal{W}}^{(n)} + \sum_{i\in\{x,y,z\}}\left(2\,\text{nnz}(\Pi_i) + 3\,\text{nnz}(A_{\Pi_i})\,\widetilde{\mathcal{C}}_{\mathcal{W}}^{(i)}\right)}{\text{nnz}(A)}, \tag{5.2}$$

where $\widetilde{\mathcal{C}}_{\mathcal{W}}^{(n)}$ and $\widetilde{\mathcal{C}}_{\mathcal{W}}^{(i)}$ are the weighted complexities of the multigrid preconditioners $B^{(n)}$ and B_{Π_i}, $i \in \{x, y, z\}$, respectively. This expression is justified in the appendix. As of the memory parameter, it reflects the memory usage of the auxiliary space preconditioner per nonzero entry of A and is given by

$$\mathcal{C}_{\mathcal{M}} = \frac{\text{nnz}(A) + \text{nnz}(A^{(n)})\,\widetilde{\mathcal{C}}_{\mathcal{M}}^{(n)} + \sum_{i\in\{x,y,z\}}\left(\text{nnz}(\Pi_i) + \text{nnz}(A_{\Pi_i})\,\widetilde{\mathcal{C}}_{\mathcal{M}}^{(i)}\right)}{\text{nnz}(A)},$$

where $\widetilde{\mathcal{C}}_{\mathcal{M}}^{(n)}$ and $\widetilde{\mathcal{C}}_{\mathcal{M}}^{(i)}$ are the so-called operator complexities of the multigrid preconditioners $B^{(n)}$ and B_{Π_i}, $i \in \{x, y, z\}$, respectively.

The memory and complexity parameters just defined depend on the weighted and operator complexities of the multigrid preconditioners $B^{(n)}$ and B_{Π_i}, $i \in \{x, y, z\}$, for the auxiliary systems. If these latter complexities are set to their lower bounds (that is, if $\widetilde{\mathcal{C}}_{\mathcal{M}}^{(n)} = \widetilde{\mathcal{C}}_{\mathcal{W}}^{(n)} = \widetilde{\mathcal{C}}_{\mathcal{M}}^{(i)} = \widetilde{\mathcal{C}}_{\mathcal{W}}^{(i)} = 1$), then the resulting values of the memory and complexity parameters become themselves the lower bounds; we shall denote them with $\mathcal{C}_{\mathcal{M}*}$ and $\mathcal{C}_{\mathcal{W}*}$. Such bounds depend only on the problem at hand and represent minimal requirements for the application of the auxiliary space preconditioner to this problem.

5.2. Comparison. The comparison with the auxiliary space preconditioner as defined by (5.1) is performed in the same setting as the previous numerical experiments; see Section 4 for details. In particular, for the preconditioners $B^{(n)}$ and B_{Π_i} for the Galerkin matrices $G^T AG$ and $\Pi_i^T A\Pi_i$, $i \in \{x, y, z\}$, we use here as well the `3.2.3-aca` release of the `AGMG` software package [21], which implements the standard aggregation-based AMG solver. The software is used as is, except for the aggregation whose parameters are chosen as suggested in Section 3 for the nodal aggregation scheme: we use 3 passes of pairwise aggregation and set the aggregate quality threshold value to $\overline{\kappa} = 20.0$; the reasons behind this choice are the same as in Section 3.

The convergence and complexity results for the two methods are given in Table 5.1 for various discrete boundary value problems. Note that, apart from the usual parameters, we also report the overall work to obtain the solution (per nonzero entry of A), which is assessed via $\mathcal{W} = \#it\,(\mathcal{C}_{\mathcal{W}} + 1)$, where 1 is added to account for the floating-point multiplications needed for the matrix-vector product in the Flexible Conjugate Gradient method.

Although all the problems have already been introduced, for the reader's convenience we briefly summarize their naming convention. All problems are defined on a square or a cube. The first few letters in the problem characterize the discretization mesh: `MODB` stands for the model structured brick mesh, `MODS` is for model structured simplex mesh, `SQR` is for unstructured 2D simplex mesh and `CUB` is for unstructured 3D simplex mesh; the letters `_R` or `_J` that follow indicate, respectively, that the mesh is refined near the center of the domain or that there is a jump in the coefficient μ. The name ends up with the indication of the dimensionality of the problem (`_2D` or `_3D`) and the boundary conditions (`_N` for natural and `_D` for Dirichlet). We note that for each problem three problem sizes are presented. The problems of the largest size correspond to system matrices with $2.9 \cdot 10^7 - 8 \cdot 10^7$ nonzero entries, for the medium size we have $5 \cdot 10^6 - 2 \cdot 10^7$ nonzero entries, and $6 \cdot 10^5 - 5 \cdot 10^6$ nonzero entries for the smallest problems. The matrix dimensions for consecutive sizes vary roughly by a factor of 8 for the three-dimensional problems and by a factor of 4 for the two-dimensional ones.

Comparing the results for the considered auxiliary space preconditioner (`ASP`) and Reitzinger-Schöberl multigrid method (`RS`), we note that the auxiliary space preconditioner often requires less iterations to converge, although the iteration counts of the two methods never differ by more than a factor of 2. On the other hand, the considered variant of the Reitzinger-Schöberl method always has lower complexity parameter $\mathcal{C}_{\mathcal{W}}$, that is, lower work per iteration, although the two work estimates do not differ by more than a factor of 2. Comparing the work $\mathcal{W}$ to solution, we note that although the auxiliary space preconditioner does slightly better for most of the problems, the difference with the considered Reitzinger-Schöberl

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.1

Convergence, complexity, and memory parameters for the auxiliary space preconditioner (ASP) and the considered method (RS) and iteration counts for the AMS preconditioner (AMS) from the hypre package, reported for various boundary value problems. The superscript ^{a} indicates that the results correspond to the auxiliary space preconditioner for which A_{Π_x} and A_{Π_y} are each being preconditioned with one iteration of symmetric Gauss-Seidel; the irrelevant field is then marked with $-^a$.

Problems			AMS	ASP						RS			
Name	n (10^5)	nnz/n	#it	#it	$\mathcal{C}_{\mathcal{W}}$	$\mathcal{W}$	$\mathcal{C}_{\mathcal{M}}$	$\mathcal{C}_{\mathcal{W}*}$	$\mathcal{C}_{\mathcal{M}*}$	#it	$\mathcal{C}_{\mathcal{W}}$	$\mathcal{W}$	$\mathcal{C}_{\mathcal{M}}$
	3.2	7.0	3	4^a	$-^a$	$-^a$	$-^a$	13.3	3.2	4	7.4	33.8	1.9
MODB_2D_D	12.8	7.0	3	4^a	$-^a$	$-^a$	$-^a$	13.3	3.2	4	7.5	33.9	1.9
	51.2	7.0	3	4^a	$-^a$	$-^a$	$-^a$	13.3	3.2	4	7.5	34.0	1.9
	0.2	30.2	6	9	13.2	127.7	2.9	8.9	2.1	16	8.2	147.7	1.6
MODB_J_3D_D	1.7	31.7	7	11	13.7	162.1	3.0	9.1	2.1	19	9.5	200.3	1.7
	14.4	32.3	8	12	14.1	180.9	3.0	9.1	2.1	21	10.4	240.0	1.8
	0.4	15.6	8	15	8.4	140.3	1.9	7.7	1.8	23	6.8	179.1	1.3
MODS_3D_D	4.0	16.0	9	17	8.1	155.1	1.9	7.5	1.8	27	6.8	211.6	1.3
	34.0	16.2	9	19	8.0	171.9	1.8	7.5	1.8	29	6.8	227.4	1.3
	0.5	15.7	7	11	8.0	98.8	1.8	7.5	1.8	22	6.5	164.9	1.3
MODS_3D_N	4.3	16.1	8	12	7.9	107.4	1.8	7.4	1.7	24	6.7	175.6	1.3
	35.1	16.2	8	13	8.0	116.4	1.8	7.4	1.7	26	6.7	201.3	1.3
	3.6	5.0	23	21	14.4	322.9	3.4	12.2	3.2	37	8.0	333.7	1.7
SQR_2D_N	14.3	5.0	23	22	16.0	373.5	3.5	12.2	3.2	39	8.1	353.1	1.7
	58.1	5.0	26	25	16.1	428.7	3.5	12.2	3.2	42	8.1	381.5	1.8
	9.6	5.0	18	23	15.2	373.5	3.5	12.2	3.2	41	8.0	370.4	1.7
SQR_R_2D_N	39.0	5.0	18	25	15.9	422.2	3.5	12.2	3.2	45	8.1	408.9	1.8
	158.7	5.0	24	26	15.9	439.8	3.5	12.2	3.2	49	8.1	445.5	1.8
	0.6	15.2	8	14	9.2	143.5	2.2	8.4	2.1	22	7.9	196.0	1.4
CUB_3D_D	5.7	15.8	11	16	8.9	157.7	2.1	8.0	2.0	24	8.0	215.3	1.4
	39.1	16.1	12	18	8.7	174.7	2.0	7.9	1.9	26	8.0	233.5	1.4
	0.7	15.3	13	15	8.8	146.4	2.0	7.8	1.9	21	7.4	176.0	1.4
CUB_J_3D_N	6.1	15.8	15	16	8.4	151.2	2.0	7.7	1.9	22	7.6	190.2	1.4
	40.3	16.1	19	17	8.3	158.7	2.0	7.7	1.9	23	7.8	201.9	1.4

method is never substantial. Moreover, the difference is mostly important for the "easiest" problem defined on a structured simplex mesh and less striking for more involved situations.

The results for MODB_2D_D deserve a special comment since the nonzero entries of the matrices $\Pi_x^T A \Pi_x$ and $\Pi_y^T A \Pi_y$ are then all positive. As a consequence, the coarsening of the resulting multigrid preconditioner can be slow, yielding high values for the weighted and operator complexities. However, the matrices $\Pi_x^T A \Pi_x$ and $\Pi_y^T A \Pi_y$ happen to be well conditioned in this case, and a simple symmetric Gauss-Seidel smoothing iteration can be used to precondition them. The results for the corresponding preconditioner scheme are therefore reported in Table 5.1, instead of the considered auxiliary space preconditioner.

Regarding the memory usage of the two methods, as reported through the memory parameter $\mathcal{C}_{\mathcal{M}}$, it is always substantially smaller for the considered variant of the Reitzinger-Schöberl method. The difference is even more striking if the memory other than the storage of A is considered, since then the indicator is $\mathcal{C}_{\mathcal{M}} - 1$. Most of the storage needed by the auxiliary space preconditioner actually comes from the auxiliary matrices and not from the corresponding preconditioners; this is seen from the fact that the values of $\mathcal{C}_{\mathcal{M}}$ and $\mathcal{C}_{\mathcal{M}*}$ are almost the same for most of the considered problems.

We also report in Table 5.1 the iteration counts for the AMS preconditioner from the `2.7` release of the `hypre` package [13, 14]. More precisely, we use the default AMG setting except for the `cycle_type` parameter, which is set to 14; this corresponds to the expression (5.1), except that the preconditioners $B^{(n)}$ and B_{Π_i}, $i \in \{x, y, z\}$, are now defined by the BoomerAMG solver, the forward and backward Gauss-Seidel preconditioner as represented by L^{-T} and L^{-1}, respectively, are replaced by the symmetric Gauss-Seidel version, and the various components are used in a slightly different order (the two first factors in (5.1) are permuted, as well as the two last ones). The overall setting is the same as before and the iteration count is that of the preconditioned Conjugate Gradient method from the `hypre` package.

Comparing the results for the considered auxiliary space preconditioner and the AMS preconditioner, we note that this latter solver often needs less iterations to converge, the ratio between the iteration counts mostly staying between 1 and 2. This is, however, compensated by the fact that the AMS solver uses the symmetric Gauss-Seidel smoother as a top-level smoothing iteration and in the multigrid preconditioners $B^{(n)}$ and B_{Π_i}, $i \in \{x, y, z\}$, instead of the forward/backward Gauss-Seidel for the considered auxiliary space preconditioner, which means that the AMS preconditioner requires roughly two times more work per iteration.

6. Conclusions. We have revisited the Reitzinger-Schöberl multigrid method for the solution of the discrete boundary value problem (1.2). Reitzinger-Schöberl multigrid is of aggregation type, and we proposed two modifications which are inspired from the standard state-of-the-art aggregation multigrid: the use of K-cycle multigrid and a variant of algebraic "nodal" aggregation that relies on multiple pairwise aggregation with quality control. The modifications aim at improving the convergence behavior and robustness of the resulting method. The resulting Reitzinger-Schöberl method has been observed to be robust on a set of problems defined on unstructured meshes with possible local refinements and involving possibly discontinuous coefficients μ^{-1} and β. The comparison with a state-of-the-art auxiliary space preconditioner revealed that the proposed variant of the Reitzinger-Schöberl method, although requiring slightly more work on the average compared to the auxiliary space preconditioner, needs significantly less memory to compute the solution of a given problem.

Acknowledgement. The work of the first author was partially supported by the Fonds de Recherche Scientifique-FNRS (Belgium) under Grant n° J.0084.16. The second author acknowledges the *Chaire internationale IN* program of ULB (Brussels, Belgium) for supporting his stay in Brussels.

Appendix A. Here we derive the expressions of the complexity parameter $\mathcal{C}_{\mathcal{W}}$ used in this work. More specifically, we first obtain such an expression for a generic multiplicative preconditioner. We then use it to justify the complexity parameters (4.1) and (5.2) for the Reitzinger-Schöberl multigrid preconditioner and the auxiliary space preconditioner, respectively. To be precise, we define the complexity parameter $\mathcal{C}_{\mathcal{W}}$ as the number of floating-point multiplications required per application of the corresponding preconditioner divided by the number $\mathrm{nnz}(A)$ of nonzero entries of the preconditioned system matrix.

The iteration matrix of the generic multiplicative preconditioner B considered here is given by

$$I - BA \;=\; (I - C_p A) \;\cdots\; (I - C_2 A)(I - C_1 A), \tag{A.1}$$

where the preconditioners C_i, $i = 1, \ldots, p$, are either available explicitly or defined implicitly through a combination of simpler preconditioners. An application of such a preconditioner can be written as

Algorithm 1 (application of B: $\mathbf{v} = B\mathbf{r}$)

1. $\mathbf{v}_1 = C_1\mathbf{r}$ *(application of C_1)*
2. $\mathbf{r}_1 = \mathbf{r} - A\,\mathbf{v}_1$ *(residual update)*
3. $\mathbf{v}_2 = C_2\mathbf{r}_1$ *(application of C_2)*
4. $\mathbf{r}_2 = \mathbf{r}_1 - A\,\mathbf{v}_2$ *(residual update)*

   ... ...
5. $\mathbf{v}_p = C_p\mathbf{r}_{p-1}$ *(application of C_p)*
6. $\mathbf{v} = \mathbf{v}_p + \cdots + \mathbf{v}_2 + \mathbf{v}_1$ *(final result of the preconditioner application)*

The complexity parameter $\mathcal{C}_{\mathcal{W}}$ associated with this preconditioner application is

$$\mathcal{C}_{\mathcal{W}} = \frac{\text{cost}(B)}{\text{nnz}(A)} = \frac{(p-1)\text{nnz}(A) + \sum_{i=1}^{p}\text{cost}(C_i)}{\text{nnz}(A)},$$

where $\text{cost}(C_i)$ is the number of floating-point multiplications required for the application of the preconditioner C_i. In particular, if the matrix C_i is available explicitly, then we have $\text{cost}(C_i) = \text{nnz}(C_i)$, except when the nonzero entries of C_i are given by ± 1, in which case $\text{cost}(C_i) = 0$. If C_i is expressed as a product $C_i = D_1 \cdots D_k$ or sum $C_i = D_1 + \cdots + D_k$ of matrices $D_1, \ldots, D_k$, then $\text{cost}(C_i) = \text{cost}(D_1) + \cdots + \text{cost}(D_k)$. If $C_i = L^{-1}$ or $C_i = L^{-T}$, where $L = \text{tril}(A)$ is a lower triangular part of A, then $\text{cost}(C_i) \approx \text{nnz}(A)/2$.

Considering now the Reitzinger-Schöberl *two*-grid preconditioner defined via equations (2.1) and (2.3), we note that its iteration matrix can be put in the form (A.1) provided that

$$C_1 = C_5^T = G\,(L^{(n)})^{-1}G^T, \quad C_2 = C_4^T = L^{-1}, \quad \text{and} \quad C_3 = PA_c^{-1}P^T.$$

Then, since $\text{cost}(C_1) = \text{cost}(C_5) \approx \text{nnz}(A^{(n)})/2$, $\text{cost}(C_2) = \text{cost}(C_4) \approx \text{nnz}(A)/2$, and $\text{cost}(C_3) = \text{cost}(A_c^{-1})$ (the nonzero entries of G and P are ± 1), there holds

$$\text{cost}(B) \approx 5\,\text{nnz}(A) + \text{nnz}(A^{(n)}) + \text{cost}(A_c^{-1}). \tag{A.2}$$

Regarding the *multi*grid version of the Reitzinger-Schöberl preconditioner, we recall that it is obtained from the *two*-grid method by replacing A_c^{-1} with 2 iteration of a multigrid preconditioner on the coarse level. It means that in the above expression the term $\text{cost}(A_c^{-1})$ is replaced by $2\,\text{cost}(B_c)$, where $\text{cost}(B_c)$ is again given by (A.2) on a coarser level and so on. This gives, using $A = A_1$, $A^{(n)} = A_1^{(n)}$, and $B_c = B_2$,

$$\begin{aligned}\text{cost}(B) &\approx 5\,\text{nnz}(A_1) + \text{nnz}(A_1^{(n)}) + 2\,\text{cost}(B_2)\\ &\approx \sum_{\ell=1}^{L-1} 2^{\ell-1}\left(5\,\text{nnz}(A_\ell) + \text{nnz}(A_\ell^{(n)})\right) + 2^{L-1}\text{cost}(A_L^{-1}),\end{aligned}$$

which yields (4.1) provided that we approximate $\text{cost}(A_L^{-1})$ by $\text{nnz}(A_L)$. This latter approximation is justified here since the coarsest grid size remains below 30 in all the numerical experiments.

Before going further, let us reuse the above reasoning for an auxiliary result. More specifically, we consider again the Reitzinger-Schöberl multigrid preconditioner but this time without the contributions of the "nodal" smoother on all levels. The corresponding complexity parameter is then the same as the complexity of a standard aggregation-based K-cycle multigrid preconditioner. Therefore, reusing the above reasoning, one obtains the following expression for this parameter

$$\mathcal{C}_{\mathcal{W}} = \sum_{\ell=1}^{L-1} 2^{\ell-1}\,\frac{3\,\text{nnz}(A_\ell)}{\text{nnz}(A)} + 2^{L-1}\frac{\text{cost}(A_L^{-1})}{\text{nnz}(A)} \approx 3\left(\sum_{\ell=1}^{L} 2^{\ell-1}\,\frac{\text{nnz}(A_\ell)}{\text{nnz}(A)}\right) =: 3\,\widetilde{\mathcal{C}}_{\mathcal{W}},$$

where $\widetilde{\mathcal{C}}_{\mathcal{W}}$ is the so-called weighted complexity of the aggregation-based multigrid preconditioner.

Considering now the auxiliary space preconditioner, we note that its iteration matrix (2.1) can be put in the form (A.1) by setting

$$C_1 = C_5 = GB^{(n)}G^T, \quad C_2 = C_4^T = L^{-1}, \quad \text{and} \quad C_3 = \sum_{i\in\{x,y,z\}} \Pi_i B_{\Pi_i} \Pi_i^T.$$

Further, we have $\text{cost}(C_1) = \text{cost}(C_5) = \text{cost}(B^{(n)})$ (the nonzero entries of G are ± 1), $\text{cost}(C_2) = \text{cost}(C_4) \approx \text{nnz}(A)/2$, and $\text{cost}(C_3) = \sum_{i\in\{x,y,z\}} 2\,\text{nnz}(\Pi_i) + \text{cost}(B_{\Pi_i})$. The identity (5.2) then follows by noting that $B^{(n)}$ and B_{Π_i}, $i = x, y, z$, are standard aggregation-based K-cycle multigrid preconditioners, for which the cost is roughly given, as discussed above, by 3 times the number of nonzero entries in the preconditioned matrix times the weighted complexity of the preconditioner.

REFERENCES

[1] D. N. ARNOLD, R. S. FALK, AND R. WINTHER, *Multigrid in $H(\,div)$ and $H(curl)$*, Numer. Math., 85 (2000), pp. 197–217.

[2] R. BECK, *Algebraic multigrid by components splitting for edge elements on simplicial triangulations*, Tech. Report SC 99–40, ZIB, Berlin, December 1999.

[3] R. BECK, P. DEUFLHARD, R. HIPTMAIR, R. HOPPE, AND B. WOHLMUTH, *Adaptive multilevel methods for edge element discretizations of Maxwell's equations*, Surveys Math. Indust., 8 (1999), pp. 271–312.

[4] P. B. BOCHEV, C. J. GARASI, J. J. HU, A. C. ROBINSON, AND R. S. TUMINARO, *An improved algebraic multigrid method for solving Maxwell's equations*, SIAM J. Sci. Comput., 25 (2003), pp. 623–642.

[5] T. BOONEN, G. DELIÉGE, AND S. VANDEWALLE, *On algebraic multigrid methods derived from partition of unity nodal prolongators*, Numer. Linear Algebra Appl., 13 (2006), pp. 105–131.

[6] A. BOSSAVIT, *Computational Electromagnetism*, Academic Press, San Diego, 1998.

[7] C. GEUZAINE AND J.-F. REMACLE, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, Internat. J. Numer. Methods Engrg., 79 (2009), pp. 1309–1331.

[8] W. HACKBUSCH, *Multigrid Methods and Applications*, Springer, Berlin, 1985.

[9] R. HIPTMAIR, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1999), pp. 204–225.

[10] R. HIPTMAIR AND J. XU, *Nodal auxiliary space preconditioning in $H(div)$ and $H(curl)$ spaces*, SIAM J. Numer. Anal., 45 (2007), pp. 2483–2509.

[11] J. J. HU, R. S. TUMINARO, P. B. BOCHEV, C. J. GARASI, AND A. C. ROBINSON, *Toward an h-independent algebraic multigrid method for Maxwell's equations*, SIAM J. Sci. Comput., 27 (2006), pp. 1669–1688.

[12] J. JONES AND B. LEE, *A multigrid method for variable coefficient Maxwell's equations*, SIAM J. Sci. Comput., 27 (2006), pp. 1689–1708.

[13] T. V. KOLEV AND P. S. VASSILEVSKI, *Some experience with a H^1-based auxiliary space AMG for $H(curl)$ problems*, Tech. Report UCRL-TR-221841, LLNL, Livermore, 2006.

[14] ———, *Parallel auxiliary space AMG for $H(curl)$ problems*, J. Comput. Math., 27 (2009), pp. 604–623.

[15] A. NAPOV, *Algebraic analysis of V-cycle multigrid and aggregation-based two-grid methods*, PhD. Thesis, Service de Métrologie Nucléaire, Université Libre de Bruxelles, Brussels, Belgium, 2010.

[16] A. NAPOV AND Y. NOTAY, *An algebraic multigrid method with guaranteed convergence rate*, SIAM J. Sci. Comput., 43 (2012), pp. A1079–A1109.

[17] A. NAPOV AND R. PERRUSSEL, *Revisiting aggregation-based multigrid for edge elements*, Technical Report, 2017. Available online at http://homepages.ulb.ac.be/~anapov.

[18] J.-C. NÉDÉLEC, *Mixed finite elements in $\mathbb{R}^3$*, Numer. Math., 35 (1980), pp. 315–341.

[19] Y. NOTAY, *Flexible conjugate gradients*, SIAM J. Sci. Comput., 22 (2000), pp. 1444–1460.

[20] Y. NOTAY, *An aggregation-based algebraic multigrid method*, Electron. Trans. Numer. Anal., 37 (2010), pp. 123–146. http://etna.ricam.oeaw.ac.at/vol.37.2010/pp123-146.dir/pp123-146.pdf

[21] Y. NOTAY, *AGMG software and documentation*, 2016. Available online at http://agmg.eu.

[22] Y. NOTAY AND A. NAPOV, *A massively parallel solver for discrete Poisson-like problems*, J. Comput. Phys., 281 (2015), pp. 237–250.

[23] Y. NOTAY AND P. S. VASSILEVSKI, *Recursive Krylov-based multigrid cycles*, Numer. Linear Algebra Appl., 15 (2008), pp. 473–487.

[24] R. PERRUSSEL, L. NICOLAS, F. MUSY, L. KRÄHENBÜHL, M. SCHATZMAN, AND C. POIGNARD, *Algebraic multilevel methods for edge elements*, IEEE Trans. Magn., 42 (2006), pp. 619–622.

[25] J. POMMIER AND Y. RENARD, *Getfem++. An open-source finite element library*. http://getfem.org.
[26] S. REITZINGER AND J. SCHÖBERL, *An algebraic multigrid method for finite element discretizations with edge elements*, Numer. Linear Algebra Appl., 9 (2002), pp. 223–238.
[27] U. TROTTENBERG, C. W. OOSTERLEE, AND A. SCHÜLLER, *Multigrid*, Academic Press, San Diego, 2001.
[28] P. S. VASSILEVSKI, *Multilevel Block Factorization Preconditioners*, Springer, New York, 2008.