SIAM J. SCI. COMPUT.
Vol. 34, No. 4, pp. A2092–A2117

A SELF-LEARNING ALGEBRAIC MULTIGRID METHOD FOR EXTREMAL SINGULAR TRIPLETS AND EIGENPAIRS*

HANS DE STERCK†

Abstract. A self-learning algebraic multigrid method for dominant and minimal singular triplets and eigenpairs is described. The method consists of two multilevel phases. In the first, multiplicative phase (setup phase), tentative singular triplets are calculated along with a multigrid hierarchy of interpolation operators that approximately fit the tentative singular vectors in a collective and self-learning manner, using multiplicative update formulas. In the second, additive phase (solve phase), the tentative singular triplets are improved up to the desired accuracy by using an additive correction scheme with fixed interpolation operators, combined with a Ritz update. A suitable generalization of the singular value decomposition is formulated that applies to the coarse levels of the multilevel cycles. The proposed algorithm combines and extends two existing multigrid approaches for symmetric positive definite eigenvalue problems to the case of dominant and minimal singular triplets. Numerical tests on model problems from different areas show that the algorithm converges to high accuracy in a modest number of iterations and is flexible enough to deal with a variety of problems due to its self-learning properties.

Key words. multilevel method, algebraic multigrid, singular values, singular vectors, eigenvalues, eigenvectors

AMS subject classifications. 65N55, 65F15

DOI. 10.1137/110823316

1. Introduction.

In this paper we present an algebraic multigrid (AMG) method for accurately computing a few of the largest or smallest singular values and associated singular vectors of a sparse rectangular matrix $A \in \mathbb{R}^{m\times n}$. Let the singular value decomposition (SVD) of A be given by

$$A = U\,\Sigma\,V^t. \tag{1.1}$$

Here, $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ with $U^t\,U = I_m$ and $V^t\,V = I_n$, where I_m and I_n are the unit matrices of sizes $m \times m$ and $n \times n$, respectively. Matrix $\Sigma \in \mathbb{R}^{m\times n}$ has the $l = \min(m, n)$ singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_l \geq 0$ of A on its diagonal. In what follows we will normally assume that $m \geq n$, except where noted otherwise. The columns u_j of U are called the left singular vectors of A, and the columns v_j of V are its right singular vectors. The n singular triplets (σ_j, u_j, v_j), $j = 1, \ldots, n$, satisfy

$$A\,v_j = \sigma_j\,u_j \qquad \text{and} \qquad A^t\,u_j = \sigma_j\,v_j. \tag{1.2}$$

For the special case that A is square and symmetric positive definite (SPD), the SVD of A coincides with the eigendecomposition of A, and a suitably simplified version of the AMG method we propose in this paper will be applicable to the problem of computing a few of the largest or smallest eigenvalues and associated eigenvectors of an SPD matrix A.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section February 4, 2011; accepted for publication (in revised form) May 15, 2012; published electronically July 19, 2012. This work was supported by NSERC of Canada and was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.

http://www.siam.org/journals/sisc/34-4/82331.html

†Department of Applied Mathematics, University of Waterloo, Waterloo, ON, Canada (hdesterck@uwaterloo.ca).

For definiteness, we will frame the presentation in most of the paper in terms of calculating a few of the singular triplets with largest singular values (which we call dominant triplets), and we will comment on the case of the singular triplets with the smallest singular values (which we call minimal triplets) at the end of the algorithm presentation. So we assume we seek the n_b dominant singular triplets (σ_j, u_j, v_j), $j = 1, \ldots, n_b$, of A with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{n_b}$.

There are many applications in scientific computing where dominant or minimal singular triplets of large sparse matrices need to be computed; see, for example, the discussion and references in [30, 3]. We mention a few examples. Latent semantic indexing determines concepts in documents by calculating dominant singular triplets of term-document matrices [19]. Similarly, principal component analysis is used in exploratory data analysis to identify orthogonal components with maximal variance, which correspond to dominant singular triplets of the data matrix [27]. In [15], a smoothed aggregation method is described for nonsymmetric linear systems that arise from partial differential equation (PDE) discretization and which requires approximate calculation of the minimal singular triplet of the problem matrix in a setup phase of the solver. Similarly, calculating dominant or minimal eigenpairs of SPD matrices also has many applications; see, e.g., [29, 6, 25, 31, 7].

Recently, there has been extensive research on algorithms for computing a few extremal singular triplets of large sparse matrices that work directly on (1.2) and only require matrix-vector multiplications with A and A^t; see, for example, [32, 30, 3] and the numerous references therein. In particular, the Lanczos bidiagonalization methods have received significant attention. More traditional approaches for computing extremal singular triplets of a matrix A proceed by applying symmetric eigenvalue solvers to $A^t A$ or to the augmented operator

$$X = \begin{bmatrix} 0 & A \\ A^t & 0 \end{bmatrix}. \tag{1.3}$$

In most solver environments, singular triplets for large sparse matrices are indeed computed by applying symmetric eigenvalue solvers to the augmented operator X of (1.3). For example, MATLAB's SVDS uses EIGS on X. General-purpose eigenvalue packages like Anasazi [4] or SLEPc [17] can be used in a similar manner for SVD computations. Standard eigenvalue approaches for large sparse symmetric matrices often rely on Lanczos techniques, for example, the implicitly restarted shift-invert Lanczos method as implemented in ARPACK, which is used in MATLAB's EIGS. Even though Lanczos-based eigensolvers have reached high levels of maturity and sophistication, there have also been many promising developments in different directions; see, for example the recent overview in [17] and [2] and developments reported in [29, 4, 37, 6, 25, 31]. Many of these developments for symmetric eigensolvers involve preconditioned iterations, and some have been in the field of algebraic multigrid methods [6, 25, 31]. In particular, it has been shown that methods based on or aided by algebraic multigrid can be very effective for certain eigenproblems for which error components that are damped only weakly by fine-level relaxation processes can be represented and damped efficiently on coarser levels [6, 25, 31].

While the approach to compute singular triplets of A by applying symmetric eigensolvers to the augmented operator X is satisfactory in many cases, it is sometimes argued that methods that work directly on (1.2) without forming the equivalent symmetric eigensystem may lead to a more efficient algorithm for SVD computations under certain circumstances [17], and the merits of these direct SVD approaches are

discussed in [32, 30, 3]. In particular, preconditioned Lanczos-type methods applied to find the near-zero eigenvalues of X are problematic because of the indefiniteness of X (suitable preconditioners may not necessarily be available).

The self-learning algebraic multigrid method for dominant and minimal singular triplets proposed in this paper works directly on (1.2). On the other hand, it can also be interpreted as a symmetric eigensolver for the augmented operator X but that takes advantage of the a priori knowledge of the special block structure of X to formulate efficient block relaxations and coarsening strategies. In the practical implementation of our algorithm we also use the products A^tA and AA^t for coarsening. It appears that multilevel methods have not been explored yet for the calculation of singular triplets working directly on (1.2). This is, perhaps, not surprising, since practical AMG methods for the SPD eigenproblem are also still quite a young area [6, 25, 31, 28, 12] (with earlier work described in [8, 7, 10, 9]). It can be expected that AMG methods for extremal singular triplets will be competitive for problems in which the extremal singular values are highly clustered and the extremal singular vectors are similar to each other such that they can be represented well collectively (in group) by an interpolation operator that interpolates coarse-grid representations of the singular vectors to the fine grid. Nonsymmetric discretized second-order elliptic PDE operators are expected to have this kind of spectral decomposition. We will investigate such a problem in the numerical results section of our paper, but we think that it is also interesting to investigate the applicability and performance of our algorithm for other, more general SVD problems, and we do so in the numerical results section as well. Numerical results will also be presented for SPD eigenproblems, since our algorithm offers a new extension of previous approaches for this type of problems as well.

Algebraic multigrid was originally developed for solving sparse systems of linear equations (see [8] and references in [38] and [18]). Over the years, its applicability has been extended in several ways, including to SPD eigenvalue problems [10, 9, 6, 25, 31, 28, 12]. The AMG method we propose belongs to the class of self-learning AMG methods (we borrow this term from [34]). In these methods, a multigrid hierarchy is built with interpolation operators that are determined adaptively and iteratively over several multilevel cycles to match approximately the vectors that are of interest in the problem at hand. For linear system solvers, these are the vectors that lie close to the null-space of the matrix, and for eigenvalue problems, they are the desired eigenvectors. In our new method for singular triplets, they will be the desired singular vectors. Self-learning AMG solvers are an active area of research and have been developed for solving linear equation systems, SPD eigenproblems, and Markov chain problems; see, for example, [8, 10, 13, 14, 9, 21, 22, 5, 41, 23, 33, 34, 15, 12]. Our AMG method is also *collective* in that it strives to represent several singular vectors by a single interpolation matrix for efficiency.

The AMG method we propose for computing dominant singular triplets consists of two multilevel phases. It combines and extends existing AMG approaches for the SPD eigenproblem that were proposed by Borzi and Borzi in [6], by Kushnir, Galun, and Brandt in [31], by Kahl in [28], and by Brandt et al. in [12]. In the first, multiplicative phase (setup phase), we calculate tentative singular triplets and a multigrid hierarchy with interpolation operators that approximately fit the tentative singular vectors in a collective and self-learning manner. This phase uses power method relaxation and multiplicative coarse-grid update formulas for the tentative singular vectors. We use the bootstrap framework [10] in this phase with least-squares fitting and random initial singular vectors in a way similar to the approach described in [31] and [28, 12] for calculating minimal eigenpairs of an SPD matrix. In other related work, the setup

phase of the algorithm described in [15] calculates an approximation of the singular vectors that correspond to the smallest singular value of a square nonsymmetric matrix in a way that is less general than but similar to our multiplicative phase. In [31], great care is taken to try to make the interpolation operators highly accurate for all eigenvectors in the spirit of the exact interpolation scheme (EIS) [9], leading to an eigenvalue solver that only employs this first, multiplicative phase with accuracy limited to the accuracy by which the single interpolation operator represents each eigenvector. In our approach, however, we use generic interpolation that fits the tentative singular vectors only approximately and we employ a second, additive phase (solve phase), in which the tentative singular triplets are improved up to the desired accuracy by using an additive correction scheme with fixed interpolation operators, combined with a Ritz update. Our additive phase is similar to the approach described by Borzi and Borzi in [6] for calculating minimal eigenpairs of an SPD matrix (which itself is an extension of [7]), but in [6] standard AMG interpolation is used and there is no initial multiplicative self-learning phase. Our hybrid multiplicative-additive approach results in a new AMG method for extremal singular triplets that combines two desirable properties: it allows for high-accuracy convergence when desired, and it is flexible enough to deal efficiently with a variety of problems due to its self-learning properties. The specialization of our algorithm to the SPD eigenpair case also leads to a new extension of the AMG eigenvalue algorithms of [6], [31], and [28, 12] that has the same desirable properties.

The remainder of this paper is structured as follows. In the next section we give a description of the first phase of our singular triplet algorithm, the multiplicative setup phase. This section also introduces a suitable generalization of the SVD for formulating the coarse-level problems. Section 3 then describes the second phase of the algorithm, the additive solve phase. Section 4 describes how it can be extended and specialized to the case of square matrices, minimal singular triplets, and extremal eigenpairs of SPD matrices. Section 5 contains extensive numerical evaluation of our algorithm, and section 6 concludes.

2. AMG SVD algorithm: Multiplicative phase. In this section, we describe the first, multiplicative phase of our algorithm to compute dominant singular triplets (σ, u, v) of rectangular matrix $A \in \mathbb{R}^{m\times n}$, satisfying

$$A\,v = \sigma\,u \qquad \text{and} \qquad A^t\,u = \sigma\,v. \tag{2.1}$$

2.1. Coarse-level equations. Consider interpolation matrices P for u and Q for v with $P \in \mathbb{R}^{m\times m_c}$, $Q \in \mathbb{R}^{n\times n_c}$, and P and Q of full rank. First assume that u lies exactly in the range of P, and v in the range of Q, so

$$u = P\,u_c \qquad \text{and} \qquad v = Q\,v_c \tag{2.2}$$

for some coarse-level vectors u_c and v_c. We define coarse-level equations

$$P^t\,A\,Q\,v_c = \sigma\,P^t\,B\,P\,u_c \qquad \text{and} \qquad Q^t\,A^t\,P\,u_c = \sigma\,Q^t\,C\,Q\,v_c \tag{2.3}$$

and coarse-level operators

$$A_c = P^t\,A\,Q, \qquad B_c = P^t\,B\,P, \qquad \text{and} \qquad C_c = Q^t\,C\,Q \tag{2.4}$$

with, for the finest-level operators, $B = I_m$ and $C = I_n$. The coarse-level version of fine-level equations (2.1) is then given by

$$A_c\,v_c = \sigma\,B_c\,u_c \qquad \text{and} \qquad A_c^t\,u_c = \sigma\,C_c\,v_c. \tag{2.5}$$

The intuition behind this approach is as follows: the coarse-level equations can be expected to be useful for finding triplet (σ, u, v), since, if (σ, u, v) is a singular triplet of A and (2.2) is assumed, then (σ, u_c, v_c) is a singular triplet of A_c. So one can see that, if P and Q can be constructed such that u and v lie exactly in their respective ranges (2.2), then a coarse-level solve can give us (σ, u, v) exactly. The same reasoning applies when coarsening is repeated recursively. Note that the B_c and C_c on all recursive levels are SPD since the P and Q are chosen of full rank. We will now consider methods to build P and Q such that u and v lie in their respective ranges approximately.

2.2. Generalization of singular value problem. Coarse-level equations (2.5) are of the form

$$A\,v = \sigma\,B\,u \qquad \text{and} \qquad A^t\,u = \sigma\,C\,v \tag{2.6}$$

with B and C SPD. The coarse-level equations motivate the following generalization of the singular value decomposition.

DEFINITION 2.1 (GSVD). *The generalized singular value decomposition (GSVD) of $A \in \mathbb{R}^{m\times n}$ with respect to $B \in \mathbb{R}^{m\times m}$ and $C \in \mathbb{R}^{n\times n}$, with B and C SPD, is given by*

$$A = B\,U\,\Sigma\,V^t\,C \tag{2.7}$$

with $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, and $\Sigma \in \mathbb{R}^{m\times n}$. The columns of U are called the left generalized singular vectors, and the columns of V are called the right generalized singular vectors. They satisfy the orthogonality relations $U^t\,B\,U = I_m = U\,B\,U^t$ and $V^t\,C\,V = I_n = V\,C\,V^t$. Matrix Σ has the $l = \min(m, n)$ real nonnegative generalized singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_l \geq 0$ on its diagonal. Equations (2.6) *are called the generalized singular value problem for matrix A with respect to matrices B and C.*

It is easy to see that the generalized singular triplets (σ, u, v) of GSVD (2.7) satisfy (2.6). When $B = I_m$ and $C = I_n$, GSVD (2.7) reduces to the standard SVD.

It has to be remarked that the notion of GSVD as defined above is different from the more commonly used GSVD of $A \in \mathbb{R}^{m\times n}$ with respect to $B \in \mathbb{R}^{p\times n}$ (with $m \geq n$), as, for example, defined in [24, p. 471]. Definition 2.1 is the sense of GSVD that we need in this paper. While (2.7) is a natural generalization of the singular value decomposition and relates to it in the same way the generalized eigenvalue problem (as it is commonly defined) relates to the standard eigenvalue problem, we have not been able to find it in the literature yet. In what follows, we formulate the properties of the GSVD that are useful for the calculations to be done in our multilevel cycles. We discuss existence and uniqueness, which is important for the well-posedness of our multilevel cycles, and we explain how the GSVD can be calculated, which we will need to do on the coarsest level of our multilevel cycles.

THEOREM 2.2. *GSVD* (2.7) *has the same existence and uniqueness properties as the standard SVD.*

Proof. This follows from a simple change of variables. With

$$T = B^{1/2}\,U, \qquad W = C^{1/2}\,V, \qquad \text{and} \qquad D = B^{-1/2}\,A\,C^{-1/2}, \tag{2.8}$$

GSVD (2.7) can be rewritten as a standard SVD

$$D = T\,\Sigma\,W^t. \tag{2.9}$$

∎

This change of variables provides a first manner of computing GSVD (2.7) using standard SVD algorithms. An alternative way of computing GSVD (2.7) proceeds as

follows. Let

$$X = \begin{bmatrix} 0 & A \\ A^t & 0 \end{bmatrix}, \tag{2.10}$$

$$Y = \begin{bmatrix} B & 0 \\ 0 & C \end{bmatrix}. \tag{2.11}$$

It is clear that X is symmetric and Y is SPD, and

$$(X - \sigma\, Y)\; z = 0 \tag{2.12}$$

is a symmetric generalized eigenvalue problem of size $(m+n)\times(m+n)$ with $m+n$ real eigenvalues σ_j and associated eigenvectors $[u_j^t\; v_j^t]^t$, which can be chosen orthonormal with respect to Y. The following theorem indicates how the solutions of this generalized eigenvalue problem can be used to compute the generalized singular triplets of GSVD (2.7).

THEOREM 2.3. *Let $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times m}$, and $C \in \mathbb{R}^{n\times n}$ with B and C SPD. Let $l = \min(m,n)$. Then the generalized eigenvalue problem*

$$\left(\begin{bmatrix} 0 & A \\ A^t & 0 \end{bmatrix} - \sigma \begin{bmatrix} B & 0 \\ 0 & C \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} = 0 \tag{2.13}$$

has $m+n$ solutions with generalized eigenvalues σ and linearly independent generalized eigenvectors $[u^t\; v^t]^t \neq 0$. There are l independent solutions with $\sigma_j \geq 0$ and vectors u_j and v_j satisfying orthogonality relations $u_j^t\, B\, u_i = \delta_{i,j}$ and $v_j^t\, C\, v_i = \delta_{i,j}$ $(j = 1,\ldots,l)$. The triplets (σ_j, u_j, v_j) are the generalized singular triplets of A with respect to B and C. Furthermore, there are l independent solutions $(-\sigma_j, u_j, -v_j)$. Finally, there are $\mathrm{abs}(m-n) = m+n-2\,l$ *independent solutions with $\sigma = 0$ and either $u = 0$ and $v \in \mathrm{Null}(A)$ or $v = 0$ and $u \in \mathrm{Null}(A^t)$.*

Proof. This follows directly from the variable transformations of (2.8), which transform the generalized eigenvalue problem from (2.13) into eigenvalue problem

$$\left(\begin{bmatrix} 0 & D \\ D^t & 0 \end{bmatrix} - \sigma \begin{bmatrix} I_m & 0 \\ 0 & I_n \end{bmatrix} \right) \begin{bmatrix} t \\ w \end{bmatrix} = 0, \tag{2.14}$$

which has the properties listed in the theorem; see, for example, [24, p. 427]. ∎

A third possible way to calculate GSVD (2.7) is by solving for the left and right generalized singular vectors separately, using

$$(A^t\, B^{-1}\, A)\, v = \sigma^2\, C\, v \qquad \text{and} \qquad (A\, C^{-1}\, A^t)\, u = \sigma^2\, B\, u. \tag{2.15}$$

2.3. Bootstrap AMG V-cycles. In this section, we describe how we use the bootstrap AMG (BAMG) approach [10] to find approximations of the desired n_b dominant singular vectors and values and adaptively determine interpolation operators that approximately fit the singular vectors. We follow the approaches described in [31] and [28, 12]. For completeness and definiteness, we briefly describe all steps in the process with some details discussed in more detail in subsequent sections. Algorithm 1 gives a high-level algorithmic description of the multiplicative V-cycle.

ALGORITHM 1. **AMG_SVD_mult** multiplicative cycle (recursive).

$(U_b, V_b, \sigma_b, U_t, V_t) =$ **AMG_SVD_mult**$(U_b, V_b, \sigma_b, U_t, V_t; A, B, C)$
input/output variables: boot singular vectors U_b and V_b and boot singular values σ_b; test singular vectors U_t and V_t; matrices A, B, C

if *not on coarsest level* **then**
relax on test vectors U_t, V_t (section 2.4), μ_t times;
relax on boot vectors U_b, V_b and update boot singular values σ_b (section 2.5), μ_b times;
coarsen u and v variables (section 2.7);
compute interpolation matrices P and Q that fit U and V boot and test vectors in their ranges (section 2.8);
form coarse operators A_c, B_c, C_c (section 2.1);
restrict U_t, V_t and U_b, V_b to coarse level (section 2.3) $\rightarrow U_{t,c}, V_{t,c}$ and $U_{b,c}, V_{b,c}$
recursive call:
$(U_{b,c}, V_{b,c}, \sigma_b, U_{t,c}, V_{t,c}) =$ **AMG_SVD_mult**$(U_{b,c}, V_{b,c}, \sigma_b, U_{t,c}, V_{t,c}; A_c, B_c, C_c)$;
interpolate $U_{b,c}, V_{b,c}$ up to current fine level (section 2.1);
relax on boot vectors U_b, V_b and update boot singular values σ_b (section 2.5), μ_b times;
`// Note: in the downward part of the first cycle (i.e., before the recursive call), the boot triplets` U_b, V_b, σ_b `are not considered; the first boot triplets are generated when the coarsest level is visited for the first time, in the else block below`
else
compute n_b singular boot triplets U_b, V_b, σ_b on coarsest level by computing generalized SVD of (A, B, C) (section 2.6);
end

We first describe the initial BAMG V-cycle. We start out on the finest level by choosing n_t random test vectors for each of u and v, and we place them in the columns of U_t and V_t, respectively. We relax on the test vectors (using a few iterations of the SVD power method for (2.1) (see below)) such that components with small σ are damped and components with large σ become dominant in the test vectors. We coarsen the finest grid (see below) and determine interpolation operators P, Q, where P fits the vectors in U_t (in a least-squares sense), such that they lie approximately in the range of P, and Q fits the vectors in V_t, such that they lie approximately in the range of Q. We also build coarse-level operators A_c, B_c, and C_c according to (2.4). We then restrict the fine-level U_t and V_t (by injection) to the first coarse level and obtain coarse versions of the test vectors stored in the columns of $U_{t,c}$ and $V_{t,c}$. We relax on $U_{t,c}$ and $V_{t,c}$ with the power method applied to (2.5). The whole process of building new, coarser interpolation operators P and Q and operators A_c, B_c, C_c by restricting $U_{t,c}$ and $V_{t,c}$ is then repeated recursively, up to some coarse level where the problem is small enough for a direct GSVD calculation.

On the coarsest level, we compute n_b dominant singular triplets by a direct decomposition and store them in vector σ_b and matrices U_b and V_b. These singular triplets are the starting approximations for our desired dominant singular triplets and will be improved in this cycle and subsequent cycles. We call the singular vectors

of these triplets the boot (singular) vectors and use the subscript b to refer to them. (We distinguish these from the initially random test vectors in U_t and V_t, which are used to get the process going and sustain it but do not directly lead to the desired n_b singular triplets themselves.) We denote by σ_b a vector with n_b components that holds approximations for the dominant singular values sought.

In the upward phase of the first BAMG V-cycle, starting from the coarsest level, we recursively interpolate the boot singular vectors U_b and V_b up to the next finer level, using the interpolation operators P and Q of the current level, according to multiplicative update formulas (2.2). On each finer level, we first relax on the boot vectors using (2.5) with the singular values in σ_b fixed and then update the elements of σ_b by recalculating the Rayleigh quotient for each pair of boot vectors (see below). Note that the test vectors U_t and V_t are not used in the upward phase of the V-cycle.

This initial BAMG multiplicative V-cycle can be followed by several additional multiplicative V-cycles. In the numerical results to be reported below, we execute a fixed small number of multiplicative V-cycles (we normally choose 5); alternatively, it is also possible to implement an automatic mechanism that executes multiplicative V-cycles until stalling convergence is detected. In the downward sweep of each of these additional cycles, one relaxes U_t and V_t as in the first V-cycle. In addition, one also relaxes the U_b and V_b and improves the σ_b on each level, as in the upward sweep of the first cycle. At each level, the vectors in both U_t and U_b are used to fit P, and the vectors in both V_t and V_b to fit Q. Then A_c, B_c, and C_c are also rebuilt using the new P and Q. The upward sweeps of the additional multiplicative cycles are the same as in the initial multiplicative cycle. At the end of every V-cycle, we optionally also apply a collective Ritz projection step (see below) to improve the boot vectors U_b, V_b and singular value approximations σ_b. We do so for the numerical tests reported in section 5.

Note that in this paper we use only the simplest type of multilevel cycles, namely, V-cycles. More sophisticated cycles including W-cycles and full multigrid (FMG) cycles [18, 6, 25, 31, 28, 12] can be considered and may lead to improved results, but for simplicity we only use V-cycles here. In the following sections we will give the details of the relaxation schemes, coarsest-level solve, coarsening, and interpolation used in our BAMG cycles.

2.4. Relaxation scheme for the test vectors. Seeking dominant singular triplets, we base relaxation for the initially random test vectors on the power method applied to (2.6). On any level, given an initial u_j we solve for v_j from

$$A^t\, u_j = C\, \bar{v}_j \qquad \text{and} \qquad v_j = \bar{v}_j/(\bar{v}_j^t C \bar{v}_j)^{1/2} \tag{2.16}$$

and then for u_j from

$$A\, v_j = B\, \bar{u}_j \qquad \text{and} \qquad u_j = \bar{u}_j/(\bar{u}_j^t B \bar{u}_j)^{1/2}. \tag{2.17}$$

This can be repeated μ_t times on each level. By applying the power method, we damp the components corresponding to small singular values and obtain initial iterates that are rich in the desired dominant singular vectors and are then used to fit the interpolation operators. In practice, we solve for the new $\bar{v}_j$ and $\bar{u}_j$ in an inexact way by performing $\mu_{t,J}$ inner iteration steps of weighted Jacobi with weight ω_J. For example, for (2.16) we iterate on

$$\bar{v}_j^{(i+1)} = \bar{v}_j^{(i)} - \omega_J\, D_C^{-1}\, (C\, \bar{v}_j^{(i)} - A^t\, u_j) \tag{2.18}$$

with $\bar{v}_j^{(0)} = v_j$ initially and with the iteration index of the weighted Jacobi procedure indicated in superscript. Here, D_C is a diagonal matrix with the diagonal of the SPD matrix C on its diagonal. In the numerical results reported in section 5, we use $\omega_J = 0.7$ and $\mu_{t,J} = 1$.

2.5. Relaxation scheme for the boot vectors and update formulas for the singular values. For the boot vectors, we relax on

$$A\,v = \sigma\,B\,u + \kappa, \tag{2.19}$$

$$A^t\,u = \sigma\,C\,v + \tau. \tag{2.20}$$

(Note that in the multiplicative phase $\kappa = 0$ and $\tau = 0$ on all levels, but the additive phase will require nonvanishing κ and τ, so we already include them in the formulation here.) On any level, given an initial σ_j, u_j and v_j, we solve for a new u_j from (2.19) and then for a new v_j from (2.20). This amounts to a block Gauss–Seidel (GS) scheme for equation system (2.19)–(2.20). For dominant σs, system (2.19)–(2.20), or, equivalently,

$$(X - \sigma\,Y)\;[u^t\;v^t]^t = [\kappa^t\,\tau^t]^t \tag{2.21}$$

may be close to diagonally dominant, so this will work well in many cases. For some problems or on coarser levels, the block GS approach may not converge well and Kaczmarz relaxation [39, 31, 28, 12] (see also below) on (2.21) or its blocks may be preferable. In our block GS approach, we again approximate the solutions of (2.19)–(2.20) in an inexact way by performing $\mu_{b,J}$ inner iteration steps of weighted Jacobi. For example, for (2.19) we iterate on

$$u_j^{(i+1)} = u_j^{(i)} - \omega_J\,D_B^{-1}\,(B\,u_j^{(i)} - (A\,v_j - \kappa)/\sigma_j). \tag{2.22}$$

In the numerical results reported in section 5, we use $\mu_{b,J} = 1$. In the multiplicative phase, with $\kappa = 0$, $\tau = 0$ on all levels, we also update the σs after every outer relaxation iteration on each level. The easiest way to do this is to use the Rayleigh quotient formula

$$\sigma = \frac{u^t A v}{(u^t B u)^{1/2}\,(v^t C v)^{1/2}} \tag{2.23}$$

for each boot singular triplet, which is what we do in the numerical results presented in section 5.

2.6. Coarsest-grid solution. Each time the coarsest level is reached, we determine new approximations for the n_b coarsest-level boot triplets by direct computation of the coarsest-level GSVD, (2.7). The n_b singular triplets with the largest singular values are selected as the new boot singular triplets. In our implementation, we choose to solve generalized eigenproblem (2.13) of Theorem 2.3 using a direct eigendecomposition algorithm.

2.7. Building P and Q: Coarsening and sparsity patterns. In order to build interpolation operators P and Q, at each level, we first coarsen the sets of unknowns in $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$ by choosing a set of m_c coarse-grid variables, C_u, out of the m fine-level variables for u and by choosing a set of n_c coarse-grid variables, C_v, out of the n fine-level variables for v. The coarse variables are called coarse-grid points or C-points. The fine-level u-variables that are not selected as C-points are

called F-points and are denoted by the set F_u. Similarly, the F-points of the fine-level v-variables are denoted by F_v. Well-known algorithms from AMG are used to determine C_u and C_v and the sparsity patterns of P and Q on each level, based on the idea of strength of connection in the operator matrices A. After this coarsening process, the matrix elements of P and Q are determined using a least-squares approach in such a way that the test vectors U_t and V_t (and, after the initial cycle, the boot vectors U_b and V_b) lie approximately in the ranges of P and Q, respectively.

For the coarsening process for the u-variables, we propose to apply standard AMG coarsening methods to matrix AA^t and we base coarsening of the v-variables on A^tA. (If A is square or square and symmetric, other choices can be made (see below.))

We implement coarsening as follows. For the u-variables we employ the standard one-pass Ruge–Stueben coarsening algorithm [36] on $N = AA^t$ using the strength of connection condition

$$\begin{array}{c} \text{variable } i \text{ is strongly influenced by variable } j \\ \Updownarrow \\ |n_{i,j}| \geq \theta \sum_k |n_{i,k}| \end{array} \tag{2.24}$$

with $0 < \theta < 1$ a fixed strength parameter that may be chosen dependent on the problem. (The (i, j) matrix element of N is denoted by $n_{i,j}$.) For diagonally dominant PDE discretizations, strength is often determined relative to the largest off-diagonal element in row i, using condition $|n_{i,j}| \geq \theta \max_{k \neq i} |n_{i,k}|$. We, however, target a broader class of problem matrices and opt for strength condition (2.24), which is somewhat more general. Note, however, that the magnitude of strength parameter θ typically needs to be chosen differently in the two approaches. For the v-variables, we determine strong connections in the same way for matrix A^tA. Once the strong connections are determined, coarsening can be performed: one-pass Ruge–Stueben coarsening is executed to determine sets of C-points and F-points for the u-variables and the v-variables. Note that devising a strength of connection measure that works for a broad set of matrices is notoriously difficult (see, e.g., [9, 10, 11, 35]). The measure in (2.24) appears to work reasonably well for the numerical tests we perform below, but it is certain that problems can be found for which it will fail.

In a next step, first for the u-variables, we determine for each F-point i in F_u a coarse interpolatory set C_u^i which contains all C-points (points in C_u) that strongly influence point i according to condition (2.24) in AA^t. The coarse interpolatory sets C_v^i of the v-variable F-points are determined in the same way based on A^tA. This defines the sparsity patterns of the interpolation operators P and Q. We explain this for P, and it is analogous for Q. For each C-point in C_u with fine-level index i, we let $\alpha(i)$ be the index of point i on the coarse level. For all points i in C_u, row i in P is zero, except for $p_{i,\alpha(i)} = 1$. For all F-points i in F_u, row i in P is zero, except for matrix elements $p_{i,\alpha(k)}$, where k is an element of i's coarse interpolatory set C_u^i. The matrix elements $p_{i,j}$ of interpolation operator P are thus given by

$$\begin{array}{llll} \text{if } i \in C_u: & p_{i,j} = 1 \quad \text{if } j = \alpha(i) & \text{and} & p_{i,j} = 0 \quad \text{otherwise,} \\ \text{if } i \in F_u: & p_{i,j} \neq 0 \quad \text{if } j = \alpha(k) \text{ with } k \in C_u^i & \text{and} & p_{i,j} = 0 \quad \text{otherwise.} \end{array}$$

Basing coarsening of the u-variables and the v-variables on AA^t and A^tA, respectively, can be motivated by the observation that, on the finest level, the left singular vectors are eigenvectors of AA^t and the right singular vectors are eigenvectors of A^tA.

Moreover, AA^t and A^tA are symmetric matrices, and AMG was built for that type of matrices. In that sense, using AA^t is a natural choice for measuring connection strength between u-variables. Also, forming AA^t can be done in $O(m)$ (assuming $m \geq n$) time for large classes of sparse matrices, so it does not overly add to the cost of our method. Note also that we only use AA^t for coarsening, and not in the rest of the algorithm, so there is no deterioration in terms of condition numbers, which is a reason to avoid calculating the left singular vectors as the eigenvectors of AA^t and the right singular vectors from A^tA. Note also that (2.15) suggests basing coarsening on $A^t\,B^{-1}\,A$ and $A\,C^{-1}\,A^t$ on coarser levels rather than A^tA and AA^t, but we normally choose to ignore the B^{-1} and C^{-1} mass matrix factors to avoid the extra matrix inversion and matrix product.

For some applications, however, it may be possible to devise good coarsening schemes for u and v directly from the rectangular matrix A by considering its rows and columns (and it may be required for performance reasons). We expect, however, that the details and success of such strategies may be highly dependent on the type of problem, and direct coarsening methods for row-variables and column-variables of rectangular matrices is kept as an interesting topic of further research.

2.8. Building P and Q: Least-squares determination of interpolation weights. We use a least-squares (LS) process to determine the interpolation weights in the rows of P and Q that correspond to F-points, following the approach in [10, 31, 28, 12]. Again, we explain the process for matrix P, and it is analogous for Q. We want to fit the interpolation weights of P such that the n_t current fine-level test vectors U_t and the n_b current boot vectors U_b (except in the first cycle) lie approximately in the range of P. Let U_f hold in its columns the $n_f = n_t + n_b$ vectors to be fitted. Let u_k be the kth vector in U_f. Let $u_{k,c}$ be the coarse-level version of u_k obtained by injection, and let $u^j_{k,c}$ be its value in coarse-level point j. Also, let u^i_k be the value of u_k in fine-level point i. The weights of each F-point row in P are determined consecutively using independent LS fits. Consider a fixed F-point with fine-level index i (the row index of P). Its coarse interpolatory set is C^i_u, and we assume now that the points in C^i_u are labeled by their coarse-level indices (the column indices of P). Let $n_{c,i}$ be the number of elements of C^i_u. For each F-point i we solve the following LS problem to determine the unknown interpolation weights $p_{i,j}$:

$$u^i_k = \sum_{j \in C^i_u} p_{i,j} u^j_{k,c} \quad (k = 1, \ldots, n_f). \tag{2.25}$$

This is a system of n_f equations in $n_{c,i}$ unknowns. We make this system overdetermined in all cases by choosing the number of initially random test vectors, n_t, larger than the expected largest interpolation stencil size $n_{c,i}$ for any i on any level. (This is one of the criteria guiding the choice of n_t, and, in our implementation, estimating n_t too small initially may require a restart of the method with a larger n_t.) Since we would like the dominant boot vectors to be fitted preferentially as soon as they become reasonable approximations, we weight the kth equation in (2.25) by the Rayleigh quotient, (2.23), of the pair (u_k, v_k); see also [31, 28, 12]. In our implementation, we solve the LS problem using a standard normal equation approach. Finally, we mention that we use a modification of (2.25) for the case of minimal singular triplets or eigenpairs, as proposed in [34]. For these cases, interpolation weights and convergence can be improved significantly by applying an extra fine-level Jacobi relaxation (using the operator we base strength on) to the F-point values u^i_k in (2.25) (but not the C-point values u^j_k); see [34] for further details. We have found in our numerical ex-

periments that this modification is not useful when seeking dominant singular triplets or eigenpairs.

3. AMG SVD algorithm: Additive phase. In the additive (solve) phase of our algorithm, we use fixed interpolation and coarse-level operators, namely, the operators P, Q, A_c, B_c, and C_c as they were determined on all levels in the last mutiplicative cycle and use an additive correction scheme to improve the n_b boot singular triplets that came out of the multiplicative (setup) cycle at the finest level. In each iteration of the additive phase, for each of the finest-level σ_j, u_j, v_j $(1 \le j \le n_b)$ in σ_b, U_b, V_b, we first improve u_j and v_j in a classical-type additive AMG V-cycle with σ_j fixed in the whole cycle. Then, after all the u_j and v_j have been updated using one V-cycle for each pair, we collectively improve all the σ_j, u_j, and v_j in σ_b, U_b, V_b using a Ritz projection step on the finest level. These multigrid-Ritz iterations are repeated until the desired accuracy is reached. Our solve phase is similar to the approach described by Borzi and Borzi in [6] for calculating minimal eigenpairs of an SPD matrix using standard AMG interpolation operators. (It is also described in [31] but not combined with a multiplicative phase.) We now extend this approach to the calculation of dominant SVD triplets using the self-learned operators from the multiplicative phase of the algorithm. Algorithm 2 gives a high-level algorithmic description of one additive cycle, and Algorithm 3 describes the AMG V-cycle that improves u_j and v_j for each j.

ALGORITHM 2. **AMG_SVD_add** additive cycle.

$(U_b, V_b, \sigma_b) =$ **AMG_SVD_add**(U_b, V_b, σ_b)
input/output variables: n_b boot singular triplets stored in U_b, V_b, and σ_b
(note: it is also assumed that the hierarchy of matrices $\{A_l\}, \{B_l\}, \{C_l\}$ and interpolation operators $\{P_l\}, \{Q_l\}$ for all multigrid levels l that was generated in the last multiplicative cycle is available on levels l)

for $j = 1 : n_b$ **do**
  $(u_j, v_j) =$ **AMG_V**$(u_j, v_j, \sigma_j, \kappa_j = 0, \tau_j = 0, l = 1, A_1, B_1, C_1, P_1, Q_1)$;
end
do a Ritz projection to update U_b, V_b, σ_b (section 3.3);

3.1. Coarse-level equations. In the additive correction scheme, the equations for triplet (σ_j, u_j, v_j) on the current level are given by

$$A\,v_j - \sigma_j\,B\,u_j = \kappa_j \qquad \text{and} \qquad A^t\,u_j - \sigma_j\,C\,v_j = \tau_j, \tag{3.1}$$

where κ_j and τ_j are the residuals restricted down from the next finer level. (So $\kappa_j = 0$ and $\tau_j = 0$ on the finest level.)

The equations on the next coarser level are then

$$A_c\,v_{j,c} - \sigma_j\,B_c\,u_{j,c} = P^t\,r_j \qquad \text{and} \qquad A_c^t\,u_{j,c} - \sigma_j\,C_c\,v_{j,c} = Q^t\,s_j, \tag{3.2}$$

where r_j and s_j are the residual vectors of the first and second fine-level equations, respectively, and the coarse-grid correction equations are given by

$$u_j^{(i+1)} = u_j^{(i)} + P\,u_{j,c} \qquad \text{and} \qquad v_j^{(i+1)} = v_j^{(i)} + Q\,v_{j,c}, \tag{3.3}$$

where the superscript (i) means the ith iterate. Note that $u_{j,c}$ and $v_{j,c}$ now represent coarse-level additive errors of the fine-level quantities u_j and v_j. Rather than using

ALGORITHM 3. **AMG_V** additive V-cycle for updating (u_j, v_j) (recursive).

$(u_j, v_j) =$ **AMG_V**$(u_j, v_j, \sigma_j, \kappa_j, \tau_j, l, A, B, C, P, Q)$
input/output variables: vectors u_j, v_j and fixed singular value σ_j;
right-hand side vectors κ_j, τ_j; multigrid level l; operators A, B, C, P, Q

if *not on coarsest level* **then**
  relax on u_j, v_j with fixed σ_j and right-hand sides κ_j, τ_j (section 2.5), μ_b times;
  compute residual vectors r_j, s_j (section 3.1);
  restrict residual vectors to the coarse level to obtain $\kappa_{j,c}, \tau_{j,c}$ (section 3.1);
  recursively solve for coarse-grid error vectors:
  $(u_{j,c}, v_{j,c}) =$ **AMG_V**$(u_{j,c} = 0, v_{j,c} = 0, \sigma_j, \kappa_{j,c}, \tau_{j,c}, l+1, A_{l+1}, B_{l+1}, C_{l+1}, P_{l+1}, Q_{l+1})$
  update u_j, v_j via coarse-grid correction (section 3.1);
  relax on u_j, v_j with fixed σ_j and right-hand sides κ_j, τ_j (section 2.5), μ_b times;
else
  compute the solutions u_j, v_j of the coarsest-level error equations by a direct solve (sections 3.1, 3.2)
end

new variable names to distinguish original variables an their coarse-level errors, we follow the convention that is common in the multigrid literature [18] to refer to variables and their coarse-level errors with the same letter from the alphabet, which aids in presenting the algorithm in a recursive way. Note that for the eigenvalue solvers in [6, 31] the additive method is described in the framework of the full approximation scheme (FAS), like in the paper in which the general ideas of this approach were originally proposed [7], where the FAS framework was required because eigenvalue approximations were modified on the coarsest level of each cycle. However, in the additive methods in [6, 31], eigenvalues remain fixed for the entire additive cycle, so there is no need to use the FAS, and the simpler error equation formulation that is common in multigrid for linear operators can be used instead, which is what we do in our discussion here.

3.2. Additive V-cycles to improve the left and right singular vectors. For each of the finest-level σ_j, u_j, v_j $(1 \leq j \leq n_b)$ in σ_b, U_b, V_b, we fix σ_j and perform an additive V-cycle as follows. We relax the singular vectors u_j and v_j using (3.1) on the finest level, with the relaxation method that was described in section 2.5. We calculate the residuals κ_j and τ_j and restrict them to the next coarser level. We then choose a zero initial guess for $u_{j,c}$ and $v_{j,c}$ and relax them using coarse equations (3.2), calculate the coarse residuals, restrict them to the next coarser level, etc. This is repeated recursively up to some coarse level where the problem is small enough for a direct solve. On the coarsest level, we solve (3.2) exactly for vector $[u_{j,c}^t \, v_{j,c}^t]^t$ (as in (2.21)). To make the coarsest-level solve somewhat more robust when the operator is close to singular, one can optionally use the pseudo-inverse (calculated via the SVD) of $X - \sigma Y$ without including the component corresponding to its smallest singular value, as suggested in [41]. We do so in the numerical results presented in section 5. We then interpolate the coarsest-grid solution up, correct using (3.3), relax the corrected vectors, interpolate up again, etc., recursively until the finest level.

3.3. Ritz projection step on the finest level to improve the boot singular triplets. After carrying out one V-cycle for each of the n_b boot singular triplets, we perform a Ritz projection step as in [6, 31]. An alternative would be to update each σ_j in σ_b using the Rayleigh quotient formula (2.23). However, a collective Ritz step leads to faster overall convergence and has other important advantages. For singular values with multiplicity larger than one, it provides orthogonal singular vectors and it precludes convergence of some of the triplets in the finest-level σ_b, U_b, and V_b to spurious duplicate triplets, which may occur with the σ's updated individually according to (2.23).

The Ritz step proceeds as follows. We first orthogonalize the columns of U_b with respect to B using the QR decomposition, and we orthogonalize the columns of V_b with respect to C. (Note that $B = I_m$ and $C = I_n$ on the finest level, but, in the multiplicative phase, the Ritz procedure can in principle also be employed on coarser levels, so we prefer to give the more general equations here.) Let $\hat{U}$ and $\hat{V}$ be the orthogonalizations of U_b and V_b, and let $\mathcal{U} = \text{span}(\hat{U})$ and $\mathcal{V} = \text{span}(\hat{V})$. We seek new $u_j \in \mathcal{U}$, $v_j \in \mathcal{V}$, and σ_j $(1 \le j \le n_b)$ such that

$$\begin{aligned} \langle u, A\,v_j - \sigma_j\,B\,u_j\rangle_B &= 0 \quad \forall u \in \mathcal{U}, \\ \langle v, A^t\,u_j - \sigma_j\,C\,v_j\rangle_C &= 0 \quad \forall v \in \mathcal{V}. \end{aligned} \tag{3.4}$$

These equations express that the residuals are desired to be orthogonal (B-orthogonal and C-orthogonal, respectively) to the spaces $\mathcal{U}$ and $\mathcal{V}$ in which we seek an improved approximation. Equation (3.4) can be expressed in terms of new variables $y, y_j \in \mathbb{R}^{m_c}$ and $z, z_j \in \mathbb{R}^{n_c}$ with $u = \hat{U}\,y$, $v = \hat{V}\,z$, $u_j = \hat{U}\,y_j$, and $v_j = \hat{V}\,z_j$, as

$$\begin{aligned} \left\langle y, \hat{U}^t\,A\,\hat{V}\,z_j - \sigma_j\,\hat{U}^t\,B\,\hat{U}\,y_j \right\rangle &= 0 \quad \forall y \in \mathbb{R}^{m_c}, \\ \left\langle z, \hat{V}^t\,A^t\,\hat{U}\,y_j - \sigma_j\,\hat{V}^t\,C\,\hat{V}\,z_j \right\rangle &= 0 \quad \forall z \in \mathbb{R}^{n_c}. \end{aligned} \tag{3.5}$$

The following generalized eigenvalue problem of size $2\,n_b \times 2\,n_b$ results:

$$\left(\begin{bmatrix} 0 & \hat{U}^t\,A\,\hat{V} \\ \hat{V}^t\,A^t\,\hat{U} & 0 \end{bmatrix} - \sigma_j \begin{bmatrix} \hat{U}^t\,B\,\hat{U} & 0 \\ 0 & \hat{V}^t\,C\,\hat{V} \end{bmatrix} \right) \begin{bmatrix} y_j \\ z_j \end{bmatrix} = 0. \tag{3.6}$$

According to Theorem 2.3, the eigenvalues of (3.6) occur in pairs symmetrically about zero, and it is sufficient to consider the n_b triplets (σ_j, y_j, z_j) with the largest values for σ_j to generate new approximations $(\sigma_j, \hat{U}\,y_j, \hat{V}\,z_j)$ for the dominant singular triplets on the finest level.

Note finally that, unlike the multiplicative cycles, the multigrid-Ritz additive iterations can converge to any required accuracy, even though on each level the u_j are not exactly in the range of the Ps and the v_js are not exactly in the range of the Qs. In practice, as demonstrated in the numerical tests below, the hybrid multiplicative-additive scheme converges up to machine accuracy if desired.

3.4. Combined algorithm with multiplicative and additive cycles. Algorithm 4 describes how the multiplicative and additive cycles are combined in the overall AMG_SVD algorithm for computing singular triplets.

We briefly discuss the cost of the various cycles. In terms of relaxations, the first multiplicative cycle requires $2\,(n_t\,\mu_t + n_b\,\mu_b)$ vector relaxations on each level, and the

ALGORITHM 4. **AMG_SVD** main algorithm.

$(U_b, V_b, \sigma_b) =$ **AMG_SVD**(A, n_b, n_t)
input variables: matrix A; desired number of singular triplets n_b; number of test vectors n_t
output variables: n_b boot singular triplets stored in U_b, V_b, and σ_b

initialize U_t, V_t by generating n_t random test vector pairs;
initialize U_b, V_b, σ_b to n_b to dummy singular triplets (they are not used in the downward part of the first multiplicative cycle);
initialize B and C to identity matrices;

`// multiplicative cycles`
repeat
  $(U_b, V_b, \sigma_b, U_t, V_t) =$ **AMG_SVD_mult**$(U_b, V_b, \sigma_b, U_t, V_t; A, B, C)$
for a fixed number of times, or until convergence stagnates;
`// note: the hierarchy of matrices` $\{A_l\}, \{B_l\}, \{C_l\}$ `and interpolation operators` $\{P_l\}, \{Q_l\}$ `for all multigrid levels` l `generated in the last multiplicative cycle is stored for use in the additive cycles below`

`// additive cycles`
repeat
  $(U_b, V_b, \sigma_b) =$ **AMG_SVD_add**(U_b, V_b, σ_b)
until desired convergence;

subsequent multiplicative cycles require $2\,(n_t\,\mu_t + 2\,n_b\,\mu_b)$ vector relaxations. The additive cycles require $4\,n_b\,\mu_b$ vector relaxations on each level. In the multiplicative cycles, however, the additional cost of forming the coarse-level matrices A_c, B_c, and C_c and determining the interpolation operators P and Q is significantly larger than the relaxation cost, which makes the multiplicative cycles significantly more expensive than the additive cycles.

We conclude this section by briefly discussing the relationship of the algorithmic framework of Algorithm 4 to other methods in the literature. Most traditional multigrid schemes for linear systems $Au = f$ use additive cycles with fixed interpolation operators P determined either with the use of geometric grid information or with the use of a priori information about smooth error components (so-called standard AMG interpolation (see [8, 18])).

Recently, hybrid multiplicative-additive methods have been developed for linear systems $Au = f$ that follow the general framework of Algorithm 4 and extend the applicability of AMG methods to systems for which the smooth error components do not satisfy the standard AMG interpolation assumptions. These methods include the adaptive AMG and adaptive smoothed aggregation methods of [13, 14], which develop self-learning interpolation operators in an initial multiplicative set-up phase. Another class of methods for linear systems $Au = f$ that follow a similar hybrid multiplicative-additive approach are the BAMG methods [10, 12]. Extending and improving these methods is an active area of research [15, 16, 34].

Standalone multiplicative schemes have been used extensively in the context of Markov chains [26, 21, 22, 40], and recently these multiplicative schemes have been combined with additive cycles [5, 41, 23] in a way similar to Algorithm 4.

For SPD eigenvalue problems, most existing multigrid methods use additive schemes with interpolation operators P determined geometrically or by the standard AMG approach [7, 6, 31]. Other approaches use multiplicative schemes [8, 10, 31, 28, 12]. Combining multiplicative and additive cycles for SPD eigenvalue problems has not been explored before and is a contribution of this paper, along with applying this approach to finding dominant and minimal singular triplets.

4. AMG SVD algorithm: Specialization and extension. In this section we discuss the specialization of the dominant singular triplet algorithm for rectangular matrices to the case of square matrices and symmetric matrices (dominant eigenpairs) and its extension to the case of minimal singular triplets (and eigenpairs).

4.1. Singular triplets of square matrices. A possible simplification for square, nonsymmetric matrices is that interpolation operators P and Q could potentially be based on A or A^t; it does not appear to be necessary to form AA^t and A^tA so that cost may be saved. Interestingly, if one wants to keep square matrices on all levels, coarsening and sparsity patterns for P and Q should both be based on either A or A^t because coarsening of A and A^t may lead to different numbers of coarse grid points (except if a coarsening method is used that is symmetric). If the left and right singular vectors are expected to be very similar such that they can all be fitted with reasonable accuracy by one interpolation operator, P and Q could even be taken the same on all levels; in that case it would also hold that $B_c = C_c$ on all levels, which can be exploited for further cost savings.

4.2. Eigenpairs of symmetric matrices. In the case of symmetric matrices, the whole algorithm simplifies significantly and becomes a combination of the minimal SPD eigenpair algorithms of [6] and [31] and [28, 12] extended to dominant eigenpairs. The resulting algorithm can be formulated in terms of operators A, B, and P on all levels. This combination of a multiplicative and an additive scheme into a hybrid method for eigenpairs has the advantages that it can converge up to machine accuracy for multiple eigenvectors with one P and that it is self-learning.

4.3. Minimal singular triplets and minimal eigenpairs. With just a few small modifications, the hybrid multiplicative-additive dominant singular triplet algorithm described above can also be used to compute the n_b singular triplets with the smallest singular values. All that is required is to modify the relaxation schemes and to select the smallest singular triplets as new boot singular triplets in the coarsest-level solve of the multiplicative phase. The weights in the LS fitting of the test and boot vectors is taken as the inverse of the Rayleigh quotient; see also [31, 28, 12, 5]. For the relaxation of the n_t initially random test vectors in U_t and V_t, we iterate on (2.6) with $\sigma = 0$ using Kaczmarz relaxation (see [39, 31, 28, 12]). Richardson iteration as in [15] can be considered as another option for relaxation. For the relaxation of the n_b boot vectors in U_b and V_b, we iterate on (2.19)–(2.20) (with the small σ's from σ_b) in a block GS fashion using Kaczmarz relaxation [39, 31] for the blocks. Numerical tests show that these Kaczmarz relaxations may sometimes result in singular vector pairs that produce a negative Rayleigh quotient. We test for this and reverse the sign of one of the singular vectors if this happens. In the case of minimal eigenpairs of symmetric matrices, GS relaxation on $Ax = 0$ can be used with Kaczmarz on coarser levels; see [31, 28, 12]. In the numerical results reported below, we use Kaczmarz relaxation on all levels when seeking minimal singular triplets or eigenpairs. Note also that, since our method is self-learning, the minimal SPD eigenpair problem can in principle also be solved simply by shifting the operator such that the spectrum ends up at the other

side of the origin, and then the algorithm for dominant eigenpairs can be used (and vice versa).

5. Numerical results. In this section, we present numerical results illustrating how our proposed method performs. We discuss four different test problems that cover the different cases of rectangular matrices, square nonsymmetric matrices, and symmetric matrices.

5.1. High-order finite volume element Laplacian on unit square. In the first test problem, we seek a few extremal singular triplets of a square, nonsymmetric matrix that results from a finite volume element (FVE) discretization with quadratic polynomials of the standard Laplacian operator on the unit square with Dirichlet boundary conditions; see [42, 1]. The operator is discretized on a structured triangular grid. For this problem, the FVE method with linear polynomials gives a discretization that is exactly the same as the Galerkin finite element discretization with linear polynomials. For higher orders, however, the FVE discretization is slightly nonsymmetric.

Figure 5.1 shows convergence results for approximating the largest and smallest singular values for a matrix with $m = n = 961$ (31×31 internal grid points). We show the base-10 logarithm of the relative error in the calculated singular values

$$error = \frac{|\sigma_{exact} - \sigma_{approx}|}{\sigma_{exact}} \tag{5.1}$$

as a function of the number of V-cycles. Here, the values σ_{exact} are high-accuracy approximations obtained by MATLAB's built-in SVD algorithms. In the left panel of Figure 5.1 (largest singular values), there are 15 multiplicative (setup) cycles followed by 25 additive (solve) cycles. Note that for this simulation we have executed 15 multiplicative cycles to clearly illustrate the stalling convergence of the multiplicative

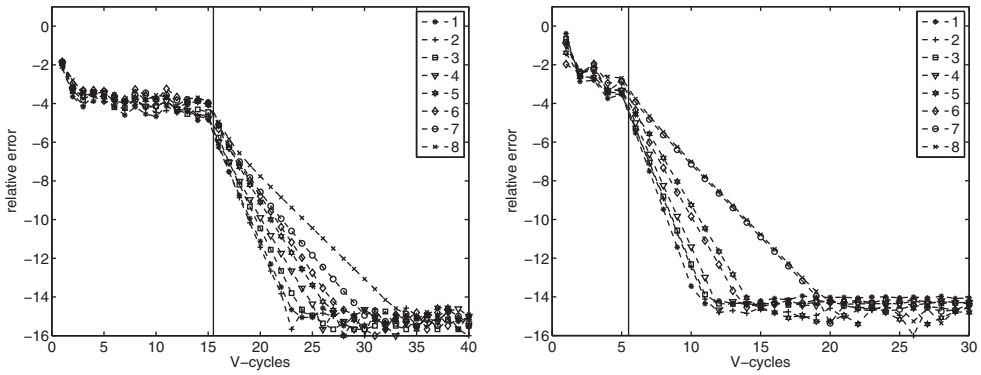

FIG. 5.1. *Largest (left panel) and smallest (right panel) singular values for high-order FVE Laplacian on unit square (square, nonsymmetric matrix). Convergence plots for the eight largest and smallest singular values (base-*10 *logarithm of relative error in singular values as a function of number of V-cycle iterations). In the left panel, singular values are labeled with decreasing magnitude (label* 1 *denotes the largest singular value). In the right panel, singular values are labeled with increasing magnitude (label* 1 *denotes the smallest singular value). The V-cycles to the left of the vertical lines are multiplicative, and the V-cycles to the right of the vertical lines are additive. For the left panel, we have executed* 15 *multiplicative cycles to clearly illustrate the stalling convergence of the multiplicative cycles. For practical application of the method, a smaller number of multiplicative cycles (for example,* 5 *or so) would suffice, as in the right panel.*

TABLE 5.1
Singular values and eigenvalues sought for each problem (high-accuracy approximations).

FVE lge	FVE sm	FD lge	FD sm	Graph lge	Graph sm	Term-Doc
7.9791546	0.01924183	7.9818877	0.01811231	13.509036	0.01000000	84.148337
7.9491729	0.04794913	7.9548012	0.04519876	13.352613	0.03456116	64.707532
7.9468326	0.04801773	7.9548012	0.04519876	13.350454	0.03901593	55.976437
7.9172573	0.07655365	7.9277148	0.07228521	12.472837	0.07966567	50.265499
7.8965349	0.09557904	7.9099298	0.09007021	12.416200	0.09490793	49.265360
7.8960066	0.09558103	7.9099298	0.09007021	11.874669	0.09918138	45.242034
7.8692955	0.12359047	7.8828433	0.11715666			44.400811
7.8616683	0.12415144	7.8828433	0.11715666			41.772394

cycles. For practical application of the method, a smaller number of multiplicative cycles would suffice, as in the right panel of Figure 5.1 (smallest singular values) where 5 multiplicative cycles are employed. We have calculated $n_b = 8$ dominant or minimal singular triplets using $n_t = 5$ initially random test vectors. We used $\mu_t = 4$ relaxations on the test vectors and $\mu_b = 4$ relaxations on the boot vectors on all levels. The coarsening strength parameter was chosen as $\theta = 0.05$. Coarsening and sparsity patterns for both P and Q are determined using A, thus guaranteeing square matrices A on all levels.

The figures show that the extremal singular triplet algorithm carries out the task that it was designed for: it collectively calculates several singular values up to machine accuracy in a modest number of multigrid V-cycles, both for the dominant triplet and the minimal triplet case. The initial, multiplicative phase approximately determines singular triplets starting from initially random test vectors, but convergence stagnates after a few operations because it is limited by the accuracy by which the singular vectors are represented collectively by single interpolation operators. A second, additive phase succeeds in driving the error to machine accuracy using the (fixed) interpolation operators that were derived in the last multiplicative iteration. This shows that the approach is able to fit interpolation to the relevant vectors both for the cases of dominant and minimal triplets.

For conciseness, we will limit ourselves to plot the relative errors in singular values or eigenvalues in this paper. Convergence of these properties goes along with high-accuracy convergence of other quantities like residuals, angles between exact and approximate singular vectors, orthogonality measures between singular vectors, etc. All these quantities also converge with high accuracy in our numerical tests, but they are not shown for conciseness. Since our code is implemented in MATLAB and is not optimized, we do not directly compare with other, optimized codes in terms of CPU time but instead focus on reporting convergence numbers as a function of numbers of V-cycle iterations, which gives valuable insight in the effectiveness of our method since the cost of a V-cycle is approximately linear in the number of unknowns, $m+n$.

For the case of dominant singular triplets (left panel of Figure 5.1), the calculation uses four levels, with coarsest size 45×45. For the case of minimal singular triplets, five levels were obtained, with a coarsest grid of size 51×51. See Table 5.1 for approximations of the singular values calculated. It can be seen that the singular values lie very close to each other, which makes this a difficult type of problem for many iterative singular value decomposition algorithms. Nevertheless, our algorithm converges to machine accuracy in a moderate number of V-cycles. Note also that the nonsymmetry of the triangulation has lifted the degeneracy of the continuous operator, which has eigenvalues with multiplicity larger than one; no singular values with multiplicity larger than one arise.

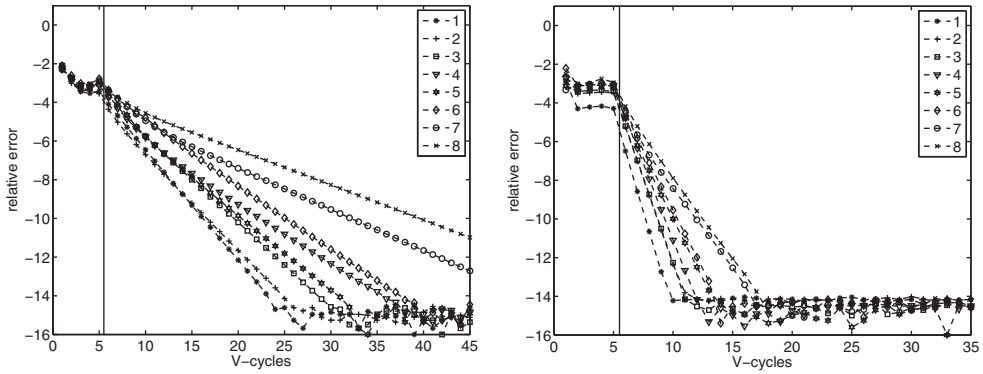

FIG. 5.2. *Largest (left panel) and smallest (right panel) eigenvalues for finite-difference Laplacian on unit square (square, symmetric matrix). In the left panel, eigenvalues are labeled with decreasing magnitude (label* 1 *denotes the largest eigenvalue). In the right panel, eigenvalues are labeled with increasing magnitude (label* 1 *denotes the smallest eigenvalue). The V-cycles to the left of the vertical lines are multiplicative, and the V-cycles to the right of the vertical lines are additive.*

An important measure for cost and scalability of AMG methods is the operator complexity. For our SVD algorithm, we define operator complexity as the sum of the number of nonzeros in operators A, B, and C on all levels divided by the number of nonzeros in A, B, and C on the finest level. (For square symmetric A, we only count the nonzeros in operators A and B.) The operator complexity for the simulations of Figure 5.1 are 3.35 (left panel) and 4.72 (right panel). Only counting the nonzeros in operators A gives complexity values of 2.51 and 3.19, respectively, so it is clear that the need to consider matrices B and C increases the operator complexity in our SVD algorithm compared to standard AMG operator complexity numbers (which only involve A).

5.2. Finite-difference Laplacian on unit square. We now consider the case of a simple finite-difference Laplacian with Dirichlet boundary conditions discretized with a 5-point stencil on a unit square with a Cartesian grid. This leads to a symmetric matrix (it is SPD), and we seek minimal and dominant eigenpairs. We use strength of connection $\theta = 0.06$ and seek $n_b = 8$ minimal or dominant eigenpairs, using $n_t = 6$ initially random test vectors. We use $\mu_t = 8$ relaxations on the test vectors and $\mu_b = 4$ relaxations on the boot vectors. We perform 5 multiplicative cycles. The problem size is $m = n = 1024$ (32×32 internal grid points). Table 5.1 shows that there are eigenvalues with multiplicity larger than one for this symmetric discretization.

The right panel of Figure 5.2 shows convergence results for the case of minimal eigenpairs. Five levels are used and the coarsest grid is of size 64×64. These results can be compared with the results of the additive-only eigenvalue method of [6] and the multiplicative-only eigenvalue methods of [31] and [28, 12]. Our additive phase is like the method in [6], but in that paper standard AMG interpolation is used. We appear to get similar results, but our method is more general and can also be applied to seeking dominant eigenpairs and to a wider set of problems due to its self-learning capacity. Our multiplicative phase is like the methods in [31] and [28, 12]. We see that convergence stagnates at the level of accuracy by which interpolation collectively represents the desired eigenvectors. In [31] interpolation is made more accurate to improve the accuracy level at which the collective multiplicative phase stagnates. As explained in that paper, the accuracy that can be obtained in this way may be sufficient for some applications, for example, due to unavoidable discretization errors

in PDE problems, or due to data and model uncertainties in data analysis tasks. In our approach, we show that, if desired, higher accuracy can be obtained by combining the multiplicative and additive approaches, resulting in a method that is flexible enough to deal efficiently with a variety of problems due to its self-learning capabilities. The left panel of Figure 5.2 gives convergence results for the case of dominant eigenpairs. Four levels are used and the coarsest grid is of size 52×52. The results show that our hybrid multiplicative-additive method can also compute dominant eigenpairs, extending the approaches for minimal eigenpairs from [6, 31, 28, 12] to dominant eigenpairs. Convergence in the additive phase appears somewhat slower than for the minimal eigenpairs case. This may be due to the fact that we employ Kaczmarz relaxation for the minimal eigenpairs, which is more efficient but also more expensive than the inexact power method relaxation used for the dominant eigenpairs case (section 2.5), or it may be due to differences in the LS weighting. It is interesting to note that the approach in [6] which uses standard AMG interpolation can also be extended to calculating dominant eigenpairs simply by changing the signs of all interpolation weights for the F-points. The resulting interpolation operators turn out to be good fits for the most oscillatory modes and can be used in an additive scheme to approximate the dominant eigenpairs. Also, in our experience, applying our algorithm with adaptive interpolation operators P to find minimal eigenpairs of the standard finite-difference Laplacian leads to interpolation operators P and multigrid convergence behavior that are almost identical to the AMG eigensolver methods of [6] with standard AMG interpolation operators. Since our method requires a few initial multiplicative V-cycles to compute the interpolation matrices which are relatively expensive, it is more efficient for this particular problem to use only additive V-cycles with standard AMG interpolation, as in [6]. Operator complexities for the simulations of Figure 5.2 when taking A and B into account are 3.13 (left panel) and 5.38 (right panel). (They are 2.29 and 3.64 when taking only A into account.)

Finally, Figure 5.3 gives convergence results for a modification of the finite-difference Laplacian test problem, namely, the advection-diffusion problem

$$-\Delta u + \sigma \vec{v} \cdot \nabla u = f \tag{5.2}$$

with $\vec{v} = (1, 2)$ and parameter σ controlling the strength of advection. Note that the advection velocity is not aligned to the grid. We discretize (5.2) with finite differences on a square Cartesian grid with 16×16 interior points and grid size parameter $h = 1$. The advection terms are discretized using the standard first-order accurate upwind discretization. This test problem is interesting because for relatively large σ it leads to highly nonsymmetric matrices that are a challenge for many multigrid algorithms. In Figure 5.3 we show convergence plots for computing minimal singular triplets of the operator of test problem (5.2) for varying parameter σ, obtained with the same algorithm settings as for the finite-difference Laplacian problem. The results show that our proposed method is able to efficiently compute minimal singular triplets for this non-grid-aligned advection-diffusion problem and for the strongly nonsymmetric matrices that are associated with strong advection. For these simulations (in order of increasing $\sigma = 0.01,\ 0.1,\ 1,\ 10$), operator complexities are 4.54, 4.18, 3.80, and 2.71 when taking A, B, and C into account. They are 2.68, 2.50, 2.33, and 2.48 when only counting the nonzeros in A.

5.3. Planar random triangulation graph Laplacian. The next test problem is the graph Laplacian operator of a planar random graph that is obtained by placing points uniformly random in the unit square and determining their Delauney

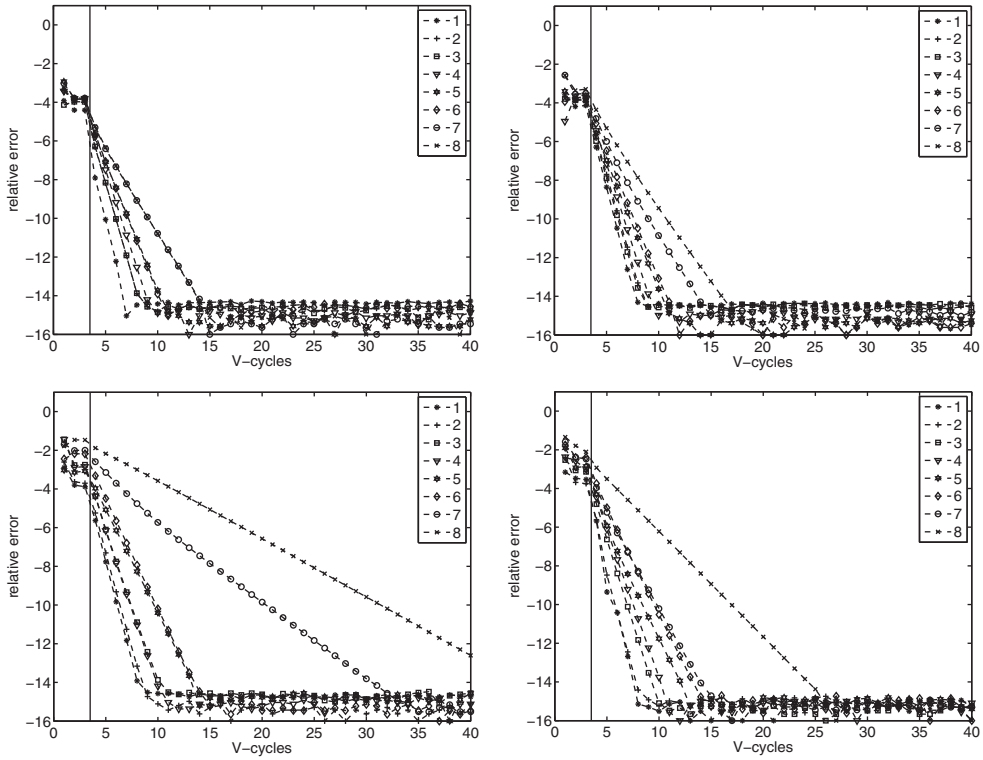

FIG. 5.3. *Smallest singular values for finite-difference advection-diffusion on square (square, nonsymmetric matrices). The advection vector is not grid-aligned, and the parameter controlling the strength of advection in* (5.2), σ, *is varied as follows:* $\sigma = 0.01$ *(top left panel),* $\sigma = 0.1$ *(top right),* $\sigma = 1$ *(bottom left), and* $\sigma = 10$ *(bottom right). Convergence plots are shown for the eight smallest singular values. Singular values are labeled with increasing magnitude (label* 1 *denotes the smallest singular value). In each plot, the* 3 *V-cycles to the left of the vertical line are multiplicative, and the* 37 *V-cycles to the right of the vertical line are additive.*

triangulation graph. With $\hat{A}$ the adjacency matrix of the graph, the graph Laplacian, A, can be constructed by setting $A = -\hat{A}$ and placing the row sums of $\hat{A}$ on the diagonal. This results in a symmetric semidefinite matrix (it has one vanishing eigenvalue), and we seek dominant and minimal eigenpairs. This problem is interesting as a test case because it is unstructured, contrary to the previous two problems which derive from structured grids. Graph Laplacian matrices are of interest in data analysis tasks [31]. We use strength of connection $\theta = 0.05$ and seek $n_b = 6$ dominant or minimal eigenpairs, using $n_t = 6$ initially random test vectors. We use $\mu_t = 1$ relaxations on the test vectors and $\mu_b = 8$ relaxations on the boot vectors. The problem size is $m = n = 1024$.

The right panel of Figure 5.4 shows convergence results for the case of minimal eigenpairs. Three levels are used and the coarsest grid is of size 59×59. The operator is shifted by 0.01 to avoid problems in representing the relative error in the smallest eigenvalue (which vanishes for the unshifted operator). Convergence behavior is satisfactory, but convergence in the additive phase is not as good as for the finite-difference Laplacian on a structured grid (Figure 5.2, right panel), even though we doubled μ_b to 8. This is most likely due to the fact that the minimal eigenvectors of the unstructured problem are less regular and less similar to each other, such that they are

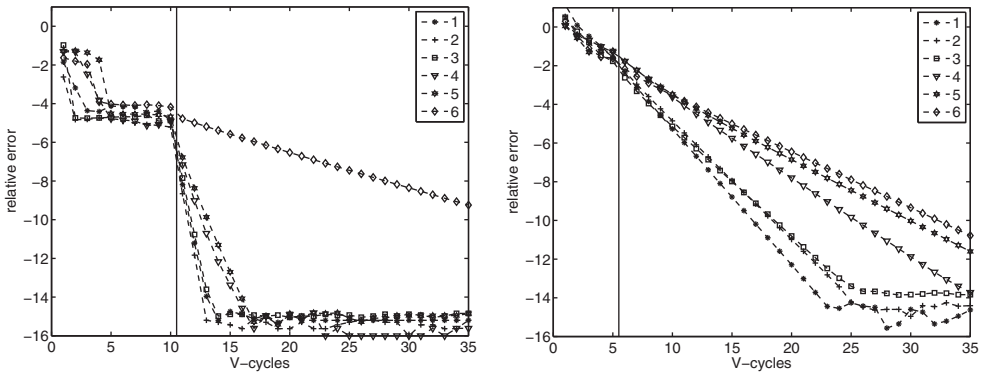

FIG. 5.4. *Largest (left panel) and smallest (right panel) eigenvalues for planar random triangulation graph Laplacian (square, symmetric matrix). In the left panel, eigenvalues are labeled with decreasing magnitude (label* 1 *denotes the largest eigenvalue). In the right panel, eigenvalues are labeled with increasing magnitude (label* 1 *denotes the smallest eigenvalue). The V-cycles to the left of the vertical lines are multiplicative, and the V-cycles to the right of the vertical lines are additive.*

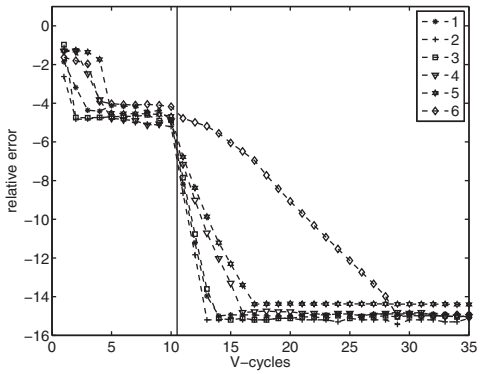

FIG. 5.5. *Same as Figure* 5.4*, left panel, but during the additive phase, whenever one or more of the eigenvalues reach a relative error converge tolerance of* 1e-14*, the interpolation operators are redetermined and preferentially fitted to the unconverged eigenpairs. This improves the convergence of the eigenpair that is slow to converge in the left panel of Figure* 5.4.

not represented as well by the collective interpolation operators. For this reason, we only sought six eigenpairs for this problem. We reduced the number of test vector relaxations because the eigenvalues are less clustered for this problem and too many test vector relaxations quickly make the set of test vectors too linearly independent for the LS fits. The operator complexity for this simulation when taking A and B into account is 1.60, and it is 1.41 when taking only A into account.

The left panel of Figure 5.4 gives convergence results for the case of dominant eigenpairs. Three levels are used and the coarsest grid is of size 77×77. The operator complexity for this simulation when taking A and B into account is 1.65, and it is 1.45 when taking only A into account. It can be seen that the algorithm converges slowly for the sixth eigenpair. When one or more of the eigenpairs sought converge significantly more slowly than the others, the following strategy can be followed to improve their convergence. In the additive phase, once some eigenpairs have converged beyond a prespecified tolerance, one can redetermine the interpolation operators in a way to preferentially fit the eigenpairs that have not converged yet. Figure 5.5 shows

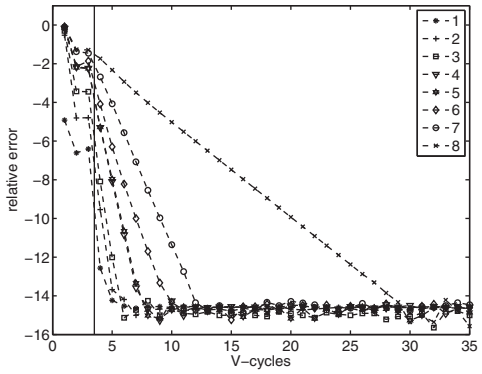

FIG. 5.6. *Largest singular values for Medline term-document matrix (rectangular). Singular values are labeled with decreasing magnitude (label* 1 *denotes the largest singular value). The* 3 *V-cycles to the left of the vertical line are multiplicative, and the* 32 *V-cycles to the right of the vertical line are additive.*

that this can improve the convergence of lagging eigenpairs. For the convergence curves shown in Figure 5.5, whenever one or more of the singular values reach a relative error converge tolerance of 1e-14, we redetermine the interpolation operators (basically, by executing one downward sweep of the multiplicative phase) and reduce the weight of the already converged boot vectors and the test vectors by a factor of 1000 in the LS fitting process. This can speed up the convergence of the remaining eigenpairs, as shown in Figure 5.5. (Note that we executed 10 multiplicative cycles for this problem because redetermining the interpolation operators in the case of just 5 initial multiplicative cycles did not lead to significant improvement.)

5.4. Medline term-document matrix. The final test matrix is a real term-document matrix, namely, the Medline data set downloaded from the Text to Matrix Generator website (http://scgroup20.ceid.upatras.gr:8000/tmg). The rows of this matrix represent terms and the columns represent documents. Matrix element (i, j) counts how many times term i occurs in document j. The matrix is sparse (less than 1% nonzeros). Latent semantic indexing determines concepts in documents by calculating dominant singular triplets of term-document matrices [19], so we seek to compute dominant singular triplets. The matrix has size $m = 5\,735$, $n = 1\,033$. We use strength of connection $\theta = 0.03$ and seek $n_b = 8$ dominant singular triplets using $n_t = 14$ initially random test vectors. We used $\mu_t = 1$ relaxations on the test vectors and $\mu_b = 4$ relaxations on the boot vectors. We perform 3 multiplicative cycles followed by 32 additive cycles.

Figure 5.6 shows convergence results for approximating the eight dominant singular triplets. The calculation uses four levels, and the coarsest grid is of size 415×198. The figure shows that our method is successful in calculating the eight dominant singular triplets with good convergence and high accuracy. The operator complexity for this simulation when taking A, B, and C into account is 2.73, and it is 2.64 when taking only A into account. The importance of this proof-of-concept calculation is that it indicates that our approach is flexible enough to deal with this kind of problem that is new to multigrid (as far as we are aware). The self-learning feature of our method is able to adapt to the singular vectors that are relevant in this application, which is interesting by itself since our development is an extension of algebraic multigrid concepts that were developed for PDEs in which the relevant vectors are of a

different nature. Similarly, we have obtained the result in Figure 5.6 using a standard PDE-oriented AMG coarsening approach and have obtained results that appear to converge quite satisfactorily. It has to be noted, though, that the dominant singular values of term-document matrices may have larger gaps (see Table 5.1), especially for the very largest ones, which may make these problems somewhat easier for iterative methods than, for example, the FVE problem of section 5.1, which has small gaps between the dominant (and minimal) singular values that decrease with increasing problem size. While we expect our method to be competitive for the latter type of problems, it remains to be investigated in future work how competitive our general approach can be made for problems like term-document matrices. For one, it would require the consideration of dedicated special-purpose coarsening methods. Forming AA^t and A^tA is unacceptable in terms of cost for most term-document matrices, and more research needs to be done to coarsen these matrices effectively. (We have already developed such special-purpose coarsening mechanisms for certain scale-free graphs; see [20] and [11, 35] for promising, more general approaches.) In the case of rectangular matrices, it may be possible to come up with methods to coarsen the row and column variables based on A and A^t directly (rather than using $A\,A^t$ and $A^t\,A$), which may be feasible for some applications, guided by the application-dependent interpretation of the variables and operator matrix coefficients; this is kept for future work. Nevertheless, the proof-of-concept results presented here already show promise and illustrate the versatility of our general approach to calculating singular triplets.

5.5. Discussion. The above numerical results show that the proposed combined multiplicative-additive approach is successful in calculating extremal singular triplets and eigenpairs with high accuracy obtained in a modest number of V-cycles for a variety of problems. However, more research needs to be done to make the method more black-box and robust. There are quite a few parameters to be chosen, and success is sometimes sensitive to the careful choice of these parameters.

For example, in the multiplicative phase, it is not always easy to find a good choice for the number of relaxations to be done on the test vectors. Too many relaxations may lead to linear dependence (and how many is too many depends on the a priori, not necessarily known gaps in the extrema of the spectrum), and not enough relaxations may lead to coarse-level problems that do not identify the correct singular triplets. Similarly, the choice of the weight factors in the LS fitting is also not straightforward and results may depend on it significantly. These aspects need to be improved. Similarly, in the additive phase, the V-cycles may not converge for some of the tentative triplets, and there is no guarantee that no triplets are missed (even though we have only rarely observed this). The current multigrid-Ritz additive phase could be replaced by methods of preconditioned inverse iteration, locally optimal block preconditioned conjugate gradient, or the Rayleigh quotient multigrid type [6, 29, 25]. Also, compatible relaxation processes may be considered for coarsening [11, 35].

6. Conclusion. We have described a new algebraic multilevel framework for computing dominant and minimal singular triplets. As far as we are aware, this is the first algebraic multigrid method that directly tackles the SVD problem without working on $A^t\,A$ or the augmented symmetric system. We combine a multiplicative phase with an additive phase to obtain a self-learning method that can converge to machine accuracy for multiple singular vectors represented collectively using single interpolation operators. The self-learning capability of the algorithm makes it applicable to many types of problems, both for dominant and minimal triplets. We have identified a generalized SVD decomposition of a matrix A relative to two SPD matrices B and C

of compatible dimensions as the problem to be solved on the coarse levels of our multilevel method and have stated its existence and uniqueness properties and discussed relevant solution methods. Our multiplicative phase follows the BAMG framework, as in [31, 28, 12] for SPD eigenproblems, and our additive phase follows a multigrid-Ritz strategy, as in [6] for SPD eigenproblems. The specialization of our combined method to SPD matrices offers a new extension of those existing AMG eigensolvers that allows for highly accurate convergence and is flexible due to its self-learning nature. Ongoing work is aimed at improving the parameter-independence and robustness of components of the algorithm, and alternative building blocks can be considered [13, 14, 6, 29, 25, 11, 35] for some of the components in the algorithmic framework. Numerical tests using our current implementation showed that convergence to high accuracy can be obtained in a modest number of V-cycles, and the versatility of the approach was illustrated by applying it to problems from different domains.

Acknowledgments. Yasunori Aoki is acknowledged for providing the FVE test matrices. The research was conducted during a sabbatical visit at the Algorithms and Complexity Department of the Max Planck Institute for Informatics in Saarbruecken, whose hospitality is greatly acknowledged.

REFERENCES

[1] Y. Aoki and H. De Sterck, *Augmented high order finite volume element method for elliptic PDEs in non-smooth domains: Convergence study*, J. Comput. Appl. Math., 235 (2011), pp. 5177–5187.

[2] P. Arbenz, U.L. Hetmaniuk, R.B. Lehoucq, and R.S. Tuminaro, *A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods*, Internat. J. Numer. Methods Engrg., 64 (2005), pp. 204–236.

[3] J. Baglama and L. Reichel, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM J. Sci. Comput., 27 (2006), pp. 19–42.

[4] C.G. Baker, U.L. Hetmaniuk, R.B. Lehoucq, and H.K. Thornquist, *Anasazi software for the numerical solution of large-scale eigenvalue problems*, ACM Trans. Math. Softw., 36 (2009), pp. 13:1–13:23.

[5] M. Bolten, A. Brandt, J. Brannick, A. Frommer, K. Kahl, and I. Livshits, *A bootstrap algebraic multilevel method for Markov chains*, SIAM J. Sci. Comput., 33 (2011), pp. 3425–3446.

[6] A. Borzi and G. Borzi, *Algebraic multigrid methods for solving generalized eigenvalue problems*, Internat. J. Numer. Methods Engrg., 65 (2006), pp. 1186–1196.

[7] A. Brandt, S. McCormick, and J. Ruge, *Multigrid methods for differential eigenproblems*, SIAM J. Sci. Statist. Comp., 4 (1983), pp. 244–260.

[8] A. Brandt, S.F. McCormick, and J.W. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1984.

[9] A. Brandt and D. Ron, *Multigrid solvers and multilevel optimization strategies*, in Multilevel Optimization and VLSICAD, J. Cong and J. R. Shinnerl, eds., Kluwer, Boston, 2003, pp. 1–69.

[10] A. Brandt, *Multiscale scientific computation: Review* 2000, in Multiscale and Multiresolution Methods: Theory and Applications, T.J. Barth, T.F. Chan, and R. Haimes, eds., Springer Verlag, Heidelberg, 2001, pp. 1–96.

[11] A. Brandt, *General highly accurate algebraic coarsening*, Electron. Trans. Numer. Anal., 10 (2000), pp. 1–20.

[12] A. Brandt, J. Brannick, K. Kahl, and I. Livshits, *Bootstrap AMG*, SIAM J. Sci. Comput., 33 (2011), pp. 612–632.

[13] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (aSA) multigrid*, SIAM Rev., 47 (2005), pp. 317–346.

[14] M. Brezina, R.D. Falgout, S. MacLachlan, T.A. Manteuffel, S.F. McCormick, and J.W. Ruge, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.

[15] M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, and G. Sanders, *Towards adaptive smoothed aggregation (aSA) for nonsymmetric problems*, SIAM J. Sci. Comput., 32 (2010), pp. 14–39.

[16] M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, G. Sanders, and P. Vassilevski, *A generalized eigensolver based on smoothed aggregation (GES-SA) for initializing smoothed aggregation (SA) multigrid*, Numer. Linear Algebra Appl., 15 (2008), pp. 249–269.

[17] V. Hernandez, J.E. Roman, A. Tomas, and V. Vidal, *A Survey of Software for Sparse Eigenvalue Problems*, SLEPc Technical Report STR-6, Universidad Politecnica de Valencia, 2009.

[18] W.L. Briggs, V. Emden Henson, and S.F. McCormick, *A Multigrid Tutorial*, SIAM, Philadelphia, 2000.

[19] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman, *Indexing by latent semantic analysis*, J. Amer. Soc. Inform. Sci., 41 (1990), pp. 391–407.

[20] H. De Sterck, V.E. Henson, and G. Sanders, *Multilevel aggregation methods for small-world graphs with application to random-walk ranking*, Comput. Inform., 30 (2011), pp. 1001–1022.

[21] H. De Sterck, T.A. Manteuffel, S.F. McCormick, K. Miller, J. Pearson, J. Ruge, and G. Sanders, *Smoothed aggregation multigrid for Markov chains*, SIAM J. Sci. Comput., 32 (2010), pp. 40–61.

[22] H. De Sterck, T.A. Manteuffel, S.F. McCormick, K. Miller, J. Ruge, and G. Sanders, *Algebraic multigrid for Markov chains*, SIAM J. Sci. Comput., 32, (2010), pp. 544–562.

[23] H. De Sterck, K. Miller, E. Treister, and I. Yavneh, *Fast multilevel methods for Markov chains*, Numer. Linear Algebra Appl., 18 (2011), pp. 961–980.

[24] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[25] U. Hetmaniuk, *A Rayleigh quotient minimization algorithm based on algebraic multigrid*, Numer. Linear Algebra Appl., 14 (2007), pp. 563–580.

[26] G. Horton and S.T. Leutenegger, *A multi-level solution algorithm for steady-state Markov chains*, in ACM SIGMETRICS Performance Evaluation Review 22, http://portal.acm.org/citation.cfm?id=183019.183040.

[27] I.T. Jolliffe, *Principal Component Analysis*, Springer, Berlin, 2002.

[28] K. Kahl, *Adaptive Algebraic Multigrid for Lattice QCD Computations*, Ph.D. thesis, University of Wuppertal, Germany, 2009.

[29] A.V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2002), pp. 517–541.

[30] E. Kokiopoulou, C. Bekas, and E. Gallopoulos, *Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization*, Appl. Numer. Math., 49 (2004), pp. 39–61.

[31] D. Kushnir, M. Galun, and A. Brandt, *Efficient multilevel eigensolvers with applications to data analysis tasks*, IEEE Trans. Pattern Anal. Machine Intelligence, 32 (2010), pp. 1377–1391.

[32] R.M. Larsen, *Lanczos Bidiagonalization with Partial Reorthogonalization*, Technical report ISSN 0105-8517, Department of Computer Science, University of Aarhus, Denmark, 1998.

[33] I. Livshits, *One-dimensional algorithm for finding eigenbasis of the Schrödinger operator*, SIAM J. Sci. Comput., 30 (2008), pp. 416–440.

[34] T. Manteuffel, S. McCormick, M. Park, and J. Ruge, *Operator-based interpolation for bootstrap algebraic multigrid*, J. Numer. Linear Algebra Appl., 17 (2010), pp. 519–537.

[35] D. Ron, R. Safro, and A. Brandt, *Relaxation based coarsening and multiscale graph organization*, Mulltiscale Model. Simul., 9 (2011), pp. 407–423.

[36] J.W. Ruge and K. Stueben, *Algebraic multigrid (AMG)*, in Multigrid Methods, Frontiers Appl. Math., S.F. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.

[37] A. Stathopoulos and J.R. McCombs, *PRIMME: Preconditioned iterative multimethod eigensolver methods and software description*, ACM Trans. Math. Softw., 37 (2010), pp. 21:1–21:30.

[38] K. Stüben, *Algebraic multigrid (AMG): An introduction with applications*, in Multigrid, U. Trottenberg, C. Oosterlee and A. Schüller, eds., Academic Press, New York, 2001.

[39] K. Tanabe, *Projection method for solving a singular system of linear equations and its applications*, Numer. Math., 17 (1971), pp. 203–214.

[40] E. Treister and I. Yavneh, *Square and stretch multigrid for stochastic matrix eigenproblems*, Numer. Linear Algebra Appl., 17 (2010), pp. 229–251.

[41] E. Treister and I. Yavneh, *On-the-fly adaptive smoothed aggregation multigrid applied to Markov chains*, SIAM J. Sci. Comput., 33 (2011), pp. 2927–2949.

[42] A. Vogel, J. Xu, and G. Wittum, *A generalization of the vertex-centered finite volume scheme to arbitrary high order*, Comput. Vis. Sci., 13 (2010), pp. 221–228.