ON MULTIPLICATIVE AND ADDITIVE MULTIGRID SCHEMES FOR SPARSE EIGENPROBLEMS

ERAN TREISTER, IRAD YAVNEH

Abstract. Multiplicative multigrid algorithms have recently been shown to be powerful and robust, mainly as pure eigen-solvers or as a setup phase for adaptive linear system solvers. However, such multiplicative algorithms have one significant drawback, in that they require recalculating the whole multigrid hierarchy of operators in each cycle, entailing a relatively high computational cost. This work suggests a scheme where multiplicative and additive algorithms are applied in a complementary manner, with the multiplicative solver providing the coarse-grid hierarchy to the additive algorithm, and getting in return an improved solution with which to construct better operators. This approach is demonstrated with two versions of recent multiplicative methods for solving Markov Chains—an improved version of Square & Stretch multigrid and an adaptive Smoothed Aggregation (αSA) algorithm.

1. Introduction

Classical algebraic multigrid (AMG) was developed and applied very successfully (mainly) to the solution of linear systems [3, 15] and eigenproblems [1, 2, 14] arising from discretization of elliptic PDEs and some types of M-Matrices. It is known, however, that classical AMG encounters difficulties when strength of connections is not easily measured. For that reason, adaptive approaches were recently adopted, especially adaptive AMG (αAMG) [5] and adaptive Smoothed Aggregation (αSA) [4], providing more general and robust linear solvers for symmetric systems. These algorithms use a combination of additive *and* multiplicative[1] schemes, where the multigrid components are built adaptively using a multilevel multiplicative setup-phase, and the linear system is then solved using the same multigrid components in a separate additive solution-phase. In the multiplicative approach, the solution itself is approximated on the coarser levels, rather than the error as in classical multigrid. This approach requires that the prolongation operator be consistently improved as the iterations progress until, ultimately, one obtains an accurate solution by prolongating a smaller vector. αSA was further developed and analyzed for nonsymmetric matrices in [6].

The multiplicative approach has also been used independently for computing the principal eigenvectors of stochastic matrices. This problem has drawn recent attention, largely due to its relevance in web search applications, among many others. Based on the classical work of [13], a whole line of recent multiplicative Markov Chain solvers were suggested in [8, 9, 10, 11, 12, 17]. Appealing as these multiplicative methods may be, they all suffer from one

Date: January 17, 2010.

Department of Computer Science, Technion—Israel Institute of Technology. eran@cs.technion.ac.il.

Research supported by the Israeli Science Foundation, grant number 795/08.

[1]In this paper "multiplicative" means that the coarse-grid correction is applied as a point-wise multiplication of the current fine approximation, while "additive" refers to the classical multigrid approach in which the coarse-grid correction is added.

fundamental common drawback—they calculate the whole multigrid hierarchy of operators in every cycle, with a computational cost that is comparable to the amount of work invested in the classical AMG setup phase. Experience has shown that in single processor environments, the setup costs of AMG (and hence—a multiplicative V-cycle) are typically comparable to several additive V-cycles [7, 16]. In [17] it was noted that, empirically, more than 50% of the computation time of each V-cycle is spent on coarse matrices construction. In parallel (distributed memory) environments, the setup phase may require significant communication between nodes and therefore a larger fraction of that total computing time.

This work follows the adaptive framework of [4, 5, 6] and suggests a new scheme which combines multiplicative and additive approaches. It aims to find the smallest (in magnitude) eigenvalue of a positive definite matrix, which is simply the null-space vector in the singular case. The scheme is demonstrated using versions of the recent Square & Stretch multigrid [17] and a version of the well-established adaptive SA of [4, 8].

We next briefly outline some concepts of additive and multiplicative multigrid techniques. Although this work focuses on a homogenous system, additive techniques are designed as linear system solvers, and therefore they will be introduced as such. On the other hand, multiplicative techniques are generally targeted at homogeneous systems, and therefore they will be introduced in that scope.

1.1. **Concepts of additive multigrid.** Additive multigrid linear system solvers generally follow a common basic idea. Given the linear system

$$A\mathbf{x} = \mathbf{b}, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is a positive definite matrix, they apply a cheap (albeit slowly converging) point-wise iterative method (a *relaxation*), such as damped *Richardson* or *Jacobi*. These methods are typically of the form:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \omega Q^{-1}(A\mathbf{x}^k - \mathbf{b}) \tag{2}$$

where ω is a scalar parameter, and the matrix Q is an easily-invertible simple preconditioner. For example, the damped-Jacobi relaxation uses the diagonal of A, $Q = \mathrm{diag}(A)$. The slow convergence of these relaxations is usually due to only a relatively small number of components in the error (called *algebraically smooth*) which approximately satisfy: $Q^{-1}A\mathbf{e} = 0$. For a simple Q, in most cases, these also approximately satisfy the homogenous system $A\mathbf{e} = 0$. For that reason, we can also refer to the *algebraically smooth* components as the *near null-space* of A. To eliminate these error components, multigrid methods use a *coarse grid correction* (CGC), applied by constructing and solving a 'coarse' system of smaller size (whose operators are denoted by a c subscript) with the aim of resolving the smooth components. Thus, the coarse grid and the relaxation fulfill complementary roles. A typical two-grid additive multigrid algorithm reads:

Additive two-grid cycle
Define $\mathbf{x}^0 = 0$ and repeat until a convergence criterion is satisfied:

(1) Apply pre-relaxations: $\mathbf{x}^k \leftarrow Relax(A, \mathbf{x}^k, \mathbf{b})$.
(2) Define the residual $\mathbf{r}^k = A\mathbf{x}^k - \mathbf{b}$ (note that $\mathbf{r}^k = A(\mathbf{x}^k - \mathbf{x}) = A\mathbf{e}^k$).
(3) Restrict the residual by a restriction operator R: $\mathbf{r}_c = R\mathbf{r}^k$.
(4) Define $\mathbf{x}_c$ as the solution of the coarse grid problem $A_c\mathbf{x}_c = \mathbf{r}_c$.
(5) Prolong $\mathbf{x}_c$ by a prolongation operator P and apply CGC: $\mathbf{x}^{k+1} = \mathbf{x}^k - P\mathbf{x}_c$.

(6) Apply post-relaxations: $\mathbf{x}^{k+1} \leftarrow \mathit{Relax}(A, \mathbf{x}^{k+1}, \mathbf{b})$.

A multi-level V-cycle is obtained by recursively treating the coarse-grid problem in step 4. For the coarse-grid operator, the Petrov-Galerkin scheme is usually employed. That is, a full rank prolongation matrix $P \in \mathbb{R}^{n \times n_c}$ (which appears in step 5) is defined, and also a suitable full rank restriction matrix $R \in \mathbb{R}^{n_c \times n}$ (which appears in step 3), where $n_c < n$ is the number of coarse-grid variables. Then,

$$A_c = RAP. \tag{3}$$

It is well-known and easy to show that if the error before the CGC is in the range of the prolongation, then the CGC eliminates it. In this work, the additive scheme will be used as a homogenous system solver, with $\mathbf{b} = 0$. Of course, on coarser levels an inhomogeneous system $A_c\mathbf{x}_c = \mathbf{r}_c$ is solved, where $\mathbf{r}_c \neq 0$.

1.2. Concepts of multiplicative multigrid.

As noted, the multiplicative multigrid technique is often targeted at a homogenous system

$$A\mathbf{x} = 0, \tag{4}$$

where A is a singular matrix. This means that there exists a non-zero vector $\mathbf{x}$ which satisfies (4). (In practice, if A is non-singular, the *near* null-space can be computed by the algorithm if we normalize the solution after each iteration. Here, we focus on the singular case.)

Suppose that we could construct some prolongation operator P, such that the solution $\mathbf{x}$ were in its range, that is, $\mathbf{x} = P\mathbf{x}_c$ for some vector $\mathbf{x}_c$ of smaller size. Defining a suitable restriction operator R, we obtain by substituting $P\mathbf{x}_c$ for $\mathbf{x}$ in equation (4) and multiplying through by R,

$$RAP\mathbf{x}_c = 0\,. \tag{5}$$

That is the definition of the coarse-grid problem, with $A_c = RAP$ the coarse-grid operator. Once solved, we obtain the sought solution by prolongation: $\mathbf{x} = P\mathbf{x}_c$. This motivates the multiplicative approach, where the prolongation P is defined such that the current solution $\mathbf{x}^k$ is in its range. As the solution becomes more and more accurate, so does the coarse representation of the original problem. With these ideas, we define the multiplicative cycle for homogeneous systems.

Multiplicative two-grid cycle

Given an initial guess $\mathbf{x}^0 \neq 0$ do until a convergence criterion is satisfied:

(1) Apply pre-relaxations: $\mathbf{x}^k \leftarrow \mathit{Relax}(A, \mathbf{x}^k)$.
(2) Construct the interpolation operator P with $\mathbf{x}^k$ in its range.
(3) Construct a restriction operator R.
(4) Calculate the coarse grid operator: $A_c = RAP$.
(5) Define $\mathbf{x}_c$ as the solution of the coarse grid problem: $A_c\mathbf{x}_c = 0$.
(6) Interpolate: $\mathbf{x}^{k+1} = P\mathbf{x}_c$.
(7) Apply post-relaxations: $\mathbf{x}^{k+1} \leftarrow \mathit{Relax}(A, \mathbf{x}^{k+1})$.

Again, a V-cycle is obtained by treating the coarse-grid problem in step 5 recursively. Here, if the fine grid solution $\mathbf{x}$ is in the range of P, then the CGC will provide the exact solution $\mathbf{x}$.

2. An additive approach for the solution of Markov Chains

2.1. **Markov Chains—problem definition.** Let $B \in R^{n\times n}$ be a given irreducible sparse column-stochastic matrix, that is, for every column j, $\sum_{i=1}^{n} B_{ij} = 1$, and all the elements of B are non–negative. A Matrix B is irreducible *iff* in its directed graph there exists a path from each vertex i to each vertex j. By the Perron–Frobenius theorem there exists a unique vector $\mathbf{x}$ with strictly positive entries which satisfies $B\mathbf{x} = \mathbf{x}$. Furthermore, $\rho(B) = 1$, where ρ denotes the spectral radius. This problem is often formulated as finding the null-vector of the *singular M-matrix*

$$A = I - B. \tag{6}$$

Our objective is to compute the principal eigenvector of B, i.e., the vector $\mathbf{x}$ which satisfies $B\mathbf{x} = \mathbf{x}$, so we seek the solution of the homogeneous system (4) with A defined in (6).

2.2. **Adaptive Solution of Linear Systems with an Additive Approach.** It is known that, when using the additive scheme of section 1.1 for solving linear systems, we must choose the prolongation P such that the algebraically smooth components are in its range. In classical AMG, an assumption is made that the near null-space of A is approximated by a locally constant vector among strong connections (typical for some types of M-Matrices), and P is constructed accordingly. Therefore, once this assumption is wrong (for example, when A is a non-uniformly scaled M-Matrix), the performance of classical AMG deteriorates [5]. To overcome this, a complete adaptive framework was developed in [4, 5] employing adaptive SA and AMG algorithms for solving symmetric linear systems. These algorithms use a combination of an additive solution-phase *and* a multiplicative setup-phase where the multigrid operators are built. The setup-phase is targeted at solving the homogeneous system (4), and it basically follows the same multiplicative scheme mentioned earlier, aiming to find the near null space of every coarse matrix at each level, and to build the transfer operators accordingly. Once the multigrid hierarchy of operators is set, the linear system is solved using these same operators in a separate standard additive solution-phase. Here we follow the adaptive framework of [4, 5], where a multiplicative setup phase and an adaptive solution phase are used. Since we are considering a homogeneous problem, both the multiplicative and additive algorithms target the same problem, so they can enhance each other.

2.2.1. *Strategies for combining additive solution and multiplicative setup.* We consider three scenarios in which additive and multiplicative multigrid schemes can be combined. The underlying assumption is that additive cycles are considerably cheaper, but they require the operators supplied by the multiplicative cycles. The first two strategies are straightforward sequential schemes, and the third aims to almost trivially exploit a slightly parallel environment of two computing units. Other strategies are worthy of consideration.

The first approach is to begin by performing a few multiplicative cycles (say, 1-4), and then switch to the additive algorithm. The second option is to run the multiplicative cycle every few additive cycles (about 4-5), updating the multigrid operators to support the next chunk of additive cycles. The third scheme is adapted from the second option and is demonstrated in a block diagram in Figure 1, where it is assumed that the multiplicative cycle is about three times as costly as the additive cycle. The first multiplicative cycle is an essential setup phase for the whole process (which may also involve tasks that do not appear in the rest of the multiplicative cycles). Once an initial hierarchy of operators is set, the additive

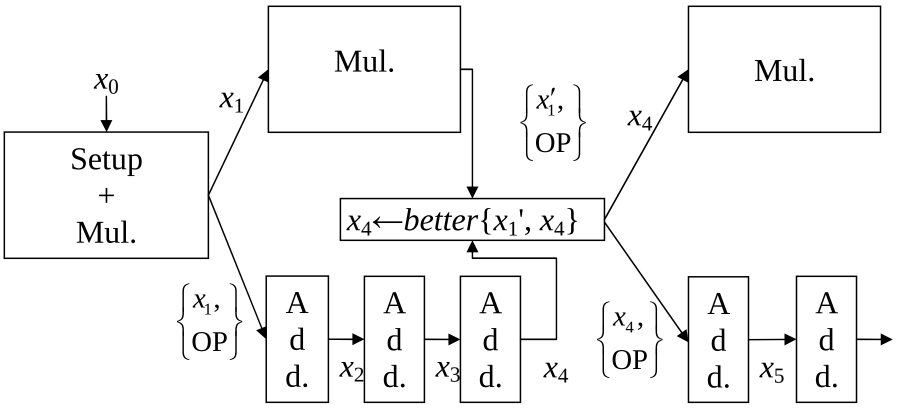

FIGURE 1. A parallel scheme for combining multiplicative and additive multigrid algorithms for finding a (near) null vector of a matrix. Multiplicative cycles marked with "Mul." and additive cycles with "Add". "OP" is a hierarchy of multigrid operators—the main output of the multiplicative cycle.

process is good to go. In parallel, the hierarchy updates can continue to evolve via the multiplicative scheme in their own (slow) pace. Once a new hierarchy is set, a cross-over of information may occur, whereby the additive solver provides the multiplicative setup process with a more accurate approximation to the solution, while the setup process provides a new hierarchy of operators to the additive solution process. Since the two processes join only at cross-over points, this parallel scheme is easy to implement on a dual-core processor, which is nowadays very common. (It does require, however, enough memory to contain two sets of coarse operator hierarchies). The same argument fits also a massively parallel scenario.

The function *better* which appears in the cross-over in Figure 1 is optional, and in our test cases is only rarely relevant (and only to the first cross-over). The point is that we have no guarantee that the operators are sufficient for the additive process to converge after only a single multiplicative cycle. For this reason, we may use this safety procedure which aims to limit unnecessary computations. In practice, we use

$$\textit{better}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \text{ if } \|A\mathbf{x}_1\| < \|A\mathbf{x}_2\|, \text{ else } \mathbf{x}_2,$$

which simply chooses the better approximation based on the residual norm. In the worst case where the additive process never works, the convergence is controlled by the multiplicative process. In cases where we only choose the additive process solution (that is, $\mathbf{x}_4$ is always chosen in Figure 1), the multiplicative cycles may terminate at the coarsest level (also noted in [4, 5]). Other choices of the function *better* might be worthy of consideration.

2.2.2. *Exact solution on the coarsest grid.* In the multiplicative cycle, the solution on the coarsest grid is done easily by calculating the null-vector directly by an eigensolver or by one iteration of the slightly shifted inverse power method. In the additive scheme however, the situation is more complicated if the matrix is singular on the coarsest grid and the system $A_c\mathbf{x}_c = \mathbf{r}_c$ has multiple solutions. If we perform a direct solution then $\mathbf{x}_c$ may have some proportion of the null-space of A_c, and assuming that P does not translate that null space to an equivalent null space of the matrix in the upper level (as in our adaptive case), this may lead to a poor CGC. On other grids, we only apply relaxations, which produce a "null-vector free" approximation. Hence, on the coarsest grid we use the pseudo inverse of A ($\mathbf{x}_c = A_c^+\mathbf{r}_c$), with a small tolerance parameter. This procedure results in a "null-vector free" solution of $A_c\mathbf{x}_c = \mathbf{r}_c$.

3. Square and Stretch—a "Smoothed Aggregation" point of view

Assume we wish to solve (4) where A is a singular positive semi-definite matrix with spectral radius ρ. Suppose that we have a restriction matrix R and a prolongation matrix P which are simple aggregation-based operators, and P has the current solution $\mathbf{x}^k$ in its range. More specifically, given the aggregates $\{\mathcal{C}_J\}_{J=1}^{n_c}$ we use the same matrices as in [17]:

$$R_{J,i} = \begin{cases} 1 & \text{if } i \in \mathcal{C}_J \\ 0 & \text{otherwise} \end{cases} \qquad P_{i,J} = \begin{cases} \mathbf{x}_i^k / \left(\mathbf{x}_c^k\right)_J & \text{if } i \in \mathcal{C}_J \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

with $\left(\mathbf{x}_c^k\right)_J = \left(R\mathbf{x}^k\right)_J = \sum_{r\in\mathcal{C}_J} \mathbf{x}_r^k$. The aggregates are constructed by the *Bottom-Up* algorithm of [17]. Naive use of the low order pure aggregation operators is inefficient, so, in the spirit of classical Smoothed Aggregation, we smooth the prolongation P with a single *Richardson* iteration matrix, with a moderate damping ρ^{-1} chosen by robustness considerations:

$$P_R = (I - \rho^{-1}A)P. \tag{8}$$

This damping is also shown to also have nice properties for complex eigenmodes in [17], where a worst-case analysis is done in the scope of the closely related weighted power method. The new coarse operator is $R(A - \rho^{-1}A^2)P$, and its spectrum bounds are much different from those of the original fine grid operator A. For example, when A is symmetric, the spectrum of the new operator is approximately bounded in the segment $[0, \frac{\rho}{4}]$, so a multiplication by 4 is needed for a recursive algorithm. If the spectrum of A is complex, a different rescaling is required so that the new spectral radius is again ρ, allowing us to use the same parameters recursively. (Equivalently, we can change the damping factor of the Richardson smoother to fit the new spectral radius.) This procedure reduces exactly to the S&S scheme of [17] if $A = I - B$, and $\rho = 2$.

3.1. Adaptive re-scaling of coarse grid operators.

Now we describe an inexpensive modification of our S&S algorithm which significantly improves its performance. Since we calculate the aggregates only once and keep them constant throughout the whole solution process, the only component that changes during the iterations is the prolongation matrix, which depends on the current solution $\mathbf{x}^k$. Because $\mathbf{x}^k$ is *locally* smooth ($A\mathbf{x}^k \approx 0$), the ratio between strongly connected values in each aggregate presumably remains approximately constant from one iteration to the next. Therefore, we expect that the spectral radii of the coarse-grid operators at corresponding levels will not vary significantly between iterations. This property clearly requires more investigation, but at least in the problems tested here (Markov Chains), it evidently holds, and it allows us to approximate the spectral radii of all operators at all levels for the whole solution process, based only on the *first* multiplicative iteration. That is, as in [17], we first find a locally smooth vector $\mathbf{x}^0$ by performing several relaxations (20 in our tests) on a positive random guess. Then we perform the first multiplicative cycle, which also involves defining the aggregates using the Bottom-Up algorithm. At that same cycle, we also approximate the spectral radius of each operator A at each level by applying m power method iteration on a random guess $\mathbf{z}_0$:

$$\mathbf{z} = A^m \mathbf{z}_0, \qquad \rho \approx \frac{\mathbf{z}^T A \mathbf{z}}{\mathbf{z}^T \mathbf{z}}\,, \tag{9}$$

which is more accurate for larger m. Since we use only a few iterations (5 in our tests), this approximation will usually provide a smaller value than the actual spectral radius (guaranteed for symmetric A). To somewhat overcome this we correct the above definition by:

$$\mathbf{z} = A^m \mathbf{z}_0, \qquad \rho \approx \min\left\{\frac{2m+2}{2m+1} \cdot \frac{\mathbf{z}^T A \mathbf{z}}{\mathbf{z}^T \mathbf{z}}, ||A||_\infty\right\} \tag{10}$$

where $||A||_\infty$, is the induced infinity matrix norm which is simply the maximal absolute sum over all rows $||A||_\infty = \max_j\{\sum_i |a_{ij}|\}$. This norm is known to usually provide a rough upper bound on the spectral radius. The factor $\frac{2m+2}{2m+1}$ is motivated by assuming a symmetric A and some probabilistic assumptions on the random vector $\mathbf{z}_0$. We will not address this choice further here. In practice, this procedure is found to yield quite an accurate approximation of the real part of the spectrum boundary and is quite consistent over different runs. In addition, choosing a moderate damping of $1/\rho$ when smoothing P also provides some safety when this estimate is too low. Note that we use the Jacobi method for relaxations, which is not affected by this re-scaling since it has a re-scaling of its own, using $diag(A)^{-1}$ as a preconditioner to A. Note that approximating the spectral radius does not much depend on the multiplicative cycle process and can be done in parallel to the cycle itself if we consider a parallel environment as in Figure 1.

To sum up, the S&S multiplicative cycle is identical to the one described in section 1.2. We use the low order restriction defined in (7), and the re-scaled prolongation

$$\frac{\tilde{\rho}(A)}{\tilde{\rho}(RAP_R)} \cdot P_R$$

where P_R is defined in (8) and $\tilde{\rho}$ is an estimated spectral radius. Note that the lumping process described in [17] may also be relevant, but we do not refer to it in this work.

4. Numerical Results

In this section we also test a version of adaptive SA. The setup phase of the method is identical to that described in section 1.2. The restriction and low order prolongation we use are identical to (7). However, we smooth the prolongation (only) by a Jacobi iteration with damping factor ω_P,

$$P_J = (I - \omega_P D^{-1} A) P\,, \tag{11}$$

where $D = \text{diag}(A)$. This method is closely related to [4, 8], where both R and P are smoothed. Smoothing only P enables us to choose smaller aggregates (using our Bottom-Up method of [17]), resulting in a smaller stencil growth during coarsening. The parameter ω_P in (11) is chosen such that the method yields good convergence and it may vary from problem to problem (choosing ω_P too big or too small may result in a deterioration of convergence). No lumping [8] is performed. In most of the examples below $\omega_P = 2/3$ was used.

We first demonstrate that the parallel strategy of Figure 1 is more efficient than the sequential approaches. We note, however, that in most of our tests an initial setup of 1-3

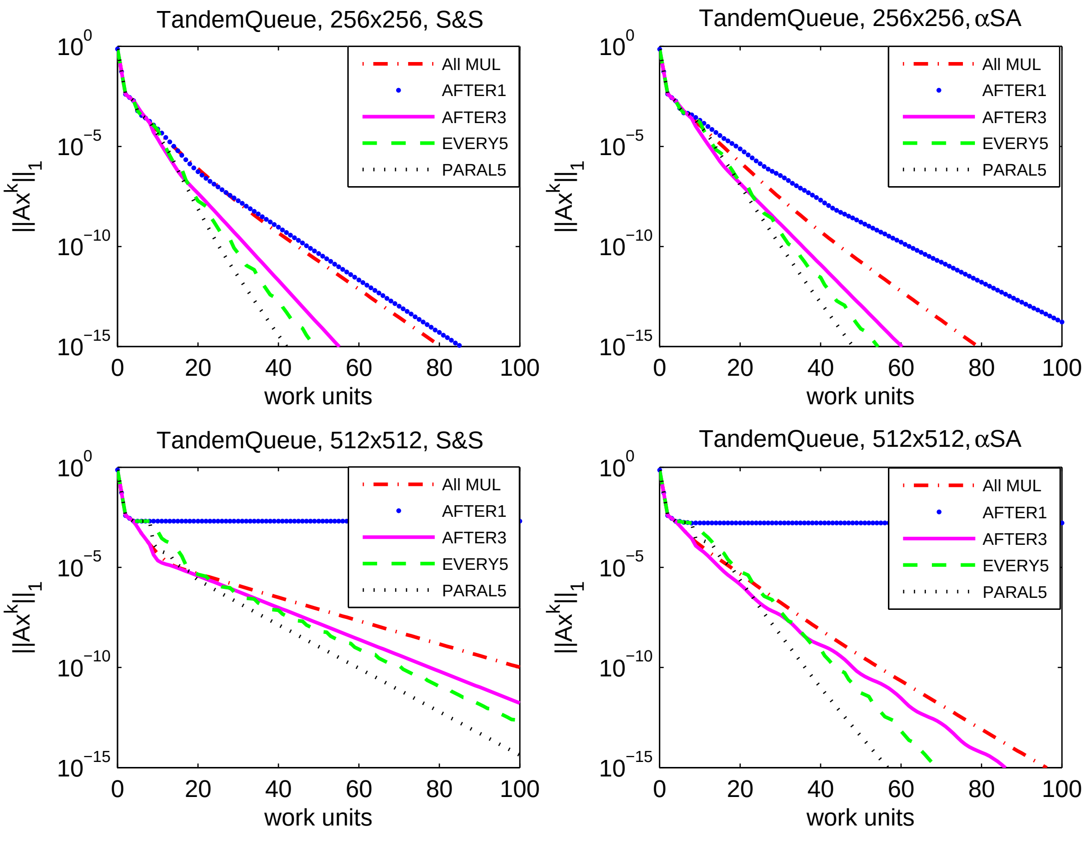

FIGURE 2. V(1,1) solution of Tandem Queue problem. Both left-hand graphs refer to S&S while the graphs on the right refer to αSA. 'All Mul' indicates a pure multiplicative solver (S&S or αSA). 'After1' and 'After3' represents an additive solution done after 1 and 3 multiplicative setup cycles respectively. 'Every5' represents performing a multiplicative cycle every 5 additive cycles (sequentially), and 'Paral5' represents a parallel scheme according to Figure 1, where a cross-over of information is done every 5 additive cycles. A multiplicative cycle is counted as two work units while an additive one is counted as 1. All aggregates and re-scaling parameters are calculated once when running 'All Mul' and kept frozen for the rest of the runs to yield a fair comparison.

multiplicative cycles suffices and yields a reasonable convergence rate. This will be investigated for more difficult problems in future work, where we expect to find more situations in which the initial setup does not suffice. Later we show some results of the S&S and αSA methods when using multiplicative and additive approaches.

In all the results we show, we start with an initial random guess and reduce the l_1 residual norm $||A\mathbf{x}^k||$ by a factor of 10^{15}. In all cases we use V(1,1) or F(1,1) cycles with one Jacobi pre-relaxation (damping: 0.7) and one Jacobi post-relaxation (damping: 0.9) in both additive and multiplicative cycles. We found these damping values to be a good choice, while the choice of the number of relaxations was not investigated thoroughly yet. In our results we try to simulate a time-wise comparison by measuring the performance by work units, where

$Problem$	n	$Method$	$\#units$	C_{op}	$\gamma : Conv.Factor$
Random	65536	αSA	99(55) [46(29)]	1.89 [3.47]	.80(.64) [.52(.33)]
Unstruct.	65536	S&S	60(37) [44(29)]	1.91 [3.52]	.65(.44) [.51(.33)]
Planar	262144	αSA	120(69)[46(29)]	1.91 [3.59]	.84(.73) [.52(.33)]
Graph	262144	S&S	64(39) [52(24)]	1.92 [3.65]	.68(.47) [.61(.33)]
	65536	αSA	72(41) [44(27)]	1.59 [2.38]	.70(.50) [.52(.33)]
Uniform	65536	S&S	70(41) [55(30)]	1.59 [2.38]	.64(.47) [.55(.35)]
2D Latice	262144	αSA	72(44) [42(29)]	1.59 [2.39]	.72(.52) [.51(.33)]
	262144	S&S	70(41) [55(30)]	1.59 [2.39]	.65(.47) [.56(.35)]
	65536	αSA	80(49) [45(37)]	1.62 [2.48]	.72(.51) [.49(.40)]
Tandem	65536	S&S	81(43) [45(37)]	1.62 [2.48]	.73(.52) [.50(.40)]
Queue	262144	αSA	98(57) [46(39)]	1.62 [2.49]	.76(.56) [.50(.39)]
	262144	S&S	180(99)[46(39)]	1.62 [2.49]	.88(.78) [.50(.40)]
	66248	αSA	82(54) [78(37)]	2.28 [4.32]	.72(.59) [.71(.41)]
Octagonal	66248	S&S	94(49) [68(43)]	2.12 [4.13]	.73(.57) [.67(.52)]
Mesh	262088	αSA	164(69)[78(42)]	2.02 [3.89]	.86(.70) [.71(.48)]
	262088	S&S	140(68)[68(43)]	2.04 [3.94]	.83(.67) [.68(.52)]

TABLE 1. Performance of multiplicative and additive solvers based on S&S and αSA. Numbers in brackets refer to V(1,1) additive cycles done according to the parallel scheme as in figure 1. The other values refer to the pure multiplicative V-cycle. Square brackets denote F(1,1) cycles. Thus, the notation is: $V_{Mul}(V_{Paral5})[F_{Mul}(F_{Paral5})]$. The convergence factor is per work unit.

each work unit stands for an additive cycle. A multiplicative cycle is counted as 2 work units (in practice it takes even more, so this is a conservative estimate). As in [17], we do 20 relaxation sweeps on an initial random guess and count that as a work unit. Then we perform the first setup phase, which also includes the aggregate definition and estimation the spectral radii (only in S&S). This is counted as 2 work units, and it appears in all settings. The operator complexity (C_{op}) and number of levels are similar in all settings since all use the same Bottom-Up algorithm with preferred aggregate size $s = 4$.

Figure 2 shows the residual norm history when solving the Tandem Queue problem presented in [8, 9, 17] by V(1,1) cycles. Here, αSA is more effective, and the results show how that neither one of the sequential strategies yields an optimal convergence of the additive method. Using more initial setup cycles will suffice at some point, however, it is hard to predict how many are required. The parallel strategy automatically performs enough setup cycles with a minimal loss of computing time.

Table 1 compares the performance of αSA and S&S in both multiplicative and additive solution schemes applied to some of the problems that appear in [9, 17]. The main measure is the asymptotic convergence factor *per work unit* (γ) and the number of work units used ($\#units$). In all cases, the convergence per unit of the additive cycles is much better than that of the multiplicative cycles. In terms of convergence, the two methods S&S and αSA are quite similar. In the first problem (random graph), S&S exhibits superior performance. This is because the parameter ω_P used in equation (11) should be different for each level since the rescaling done by the Jacobi method is far from optimal. The adaptive re-scaling performed in S&S solves this problem. On the other hand, in some problems, the adaptive calculation of the spectral radius may be inaccurate, causing slow convergence of S&S (mostly in cases of over-estimation).

Overall, the performance of both methods is usually better than that presented in [17] where 2 pre-relaxations were used (and not 1) and optimal constant stretching was chosen for each example (note that in order to compare performance of the multiplicative case, the square root of the convergence factor should be taken, since we count each cycle as one work unit). As in [17], performing F-cycles is worthwhile in most cases.

5. Conclusions and Future Work

This paper introduces the idea of employing complementing multiplicative and additive multigrid approaches for solving eigenproblems. By work unit measures, this approach is shown to be much superior to using the multiplicative approach as a stand-alone solver. A parallel strategy is suggested for combining the two approaches, where the computing time of the solution is controlled mostly by the significantly cheaper additive solution phase. This strategy seems to have potential and is worthy of further investigation. Further work is also needed for choosing the number of pre and post relaxations (and their damping factors) in both multiplicative and additive approaches. Optimizing these may improve the results shown.

The new idea is demonstrated using versions of two methods, S&S and αSA, for the solution of Markov Chains, and both methods were found applicable in the new strategy. It is very likely that other methods are applicable as well. The new version of S&S with an estimation of the spectral radii on all grids, has equal or better performance than the previous version of [17] without a need to determine the stretching parameter for each problem. The present version of αSA also shows very nice performance and is comparable to S&S. Each of these methods was found superior in different problems. Understanding this and improving them is also a subject of future research.

References

[1] A. Borzi and G. Borzi, *Algebraic multigrid methods for solving generalized eigenvalue problems*, Int. J. Numer. Meth. Eng., 65 (2006), pp. 1186–1196.

[2] A. Brandt, S. McCormick, and J. Ruge, *Multigrid methods for differential eigenproblems*, SIAM. J. Stat. Sci. Comput., 4 (1983), pp. 655–684.

[3] A. Brandt, S. F. McCormick, and J. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and its applications, D. J. Evans, ed., Cambridge University Press, Cambridge, 1984, pp. 257–284.

[4] M. Brezina, R. Falgout, S. Maclachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (α SA)*, SIAM J. Sci. Comput., 25 (2004), pp. 1896–1920.

[5] ——, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.

[6] M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, and G. Sanders, *Towards adaptive smooth aggregation (α SA) for nonsymmetric problems*, SIAM J. Sci. Comput., to appear (2009).

[7] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge, *Robustness and scalability of algebraic multigrid*, SIAM J. Sci. Comput., 21 (2000), pp. 1886–1908.

[8] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Pearson, J. Ruge, and G. Sanders, *Smoothed aggregation multigrid for markov chains*, Accepted to SIAM J. Sci. Comput., (2009).

[9] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Ruge, and G. Sanders, *Algebraic multigrid for markov chains*, Accepted to SIAM J. Sci. Comput., (2009).

[10] H. De Sterck, T. A. Manteuffel, S. F. McCormick, Q. Nguyen, and J. Ruge, *Multilevel adaptive aggregation for Markov chains, with application to web ranking*, SIAM J. Sci. Comput, 30 (2008), pp. 2235–2262.

[11] H. De Sterck, K. Miller, T. A. Manteuffel, and G. Sanders, *Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to markov chains*, Submitted to Advances in Computational Mathematics, (2009).

[12] H. De Sterck, K. Miller, G. Sanders, and M. Winlaw, *Recursively accelerated multilevel aggregation for markov chains*, Submitted to SIAM J. Sci. Comput., (2009).

[13] G. Horton and S. T. Leutenegger, *A multi-level solution algorithm for steady-state Markov chains*, Perform. Eval. Rev., 22 (1994), pp. 191–200.

[14] S. McCormick, *Multilevel adaptive methods for elliptic eigenproblems: a two-level convergence theory*, SIAM J. Numer. Anal., 31 (1994), pp. 1731–1745.

[15] J. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, frontiers in applied mathematics, S. F. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.

[16] K. Stüben, *Algebraic multigrid (amg): experiences and comparisons*, Appl. Math. Comput., 13.

[17] E. Treister and I. Yavneh, *Square and stretch multigrid for stochastic matrix eigenproblems*, Accepted to Numerical Linear Algebra with Application, (2009).