ON-THE-FLY ADAPTIVE SMOOTHED AGGREGATION MULTIGRID FOR MARKOV CHAINS[*]

ERAN TREISTER[†] AND IRAD YAVNEH[†]

Abstract. A new adaptive algebraic multigrid scheme is developed for the solution of Markov chains, where the hierarchy of operators is adapted on-the-fly in a setup process that is interlaced with the solution process. The setup process feeds the solution process with improved operators, while the solution process provides the adaptive setup process with better approximations on which to base further-improved operators. The approach is demonstrated using Petrov-Galerkin smoothed aggregation where only the prolongation operator is smoothed, while the restriction remains of low order. Results show that the on-the-fly adaptive scheme can improve the performance of multigrid solvers that require extensive setup computations, in both serial and parallel environments.

Key words. Adaptive Smoothed Aggregation, Adaptive AMG, Markov chains, Parallel Algebraic Multigrid.

1. Introduction.

Classical algebraic multigrid (AMG) was developed and applied very successfully, mainly to the solution of linear systems [4, 28] and eigenproblems [2, 3, 22] arising from discretization of elliptic partial differential equations (PDEs) and some types of M-Matrices. It is known, however, that classical AMG encounters difficulties in solving problems where the strength of connections is not easily measured. For that reason, a variant of algebraic solvers called Smoothed Aggregation (SA) was developed [35, 36, 38], in order to provide an alternative to the classical AMG. A natural generalization of smoothed aggregation is the so-called Petrov-Galerkin smoothed aggregation [12, 37] that proved useful mainly for convection-diffusion problems [13], and other nonsymmetric problems [29]. A related variant that recently received attention is the accelerated pure-aggregation method [18, 23, 25, 24, 26].

Such multigrid methods feature operators that are built based on the assumption that the constant vector locally approximates the low-energy modes (also known as the near null-space) of the matrix, or at least it is assumed that they are known. If this assumption is not satisfied, the efficiency of the methods may deteriorate. For this reason, already in [4, 27, 28] an adaptive AMG approach was suggested and self-improving AMG algorithms were outlined. The idea was to construct a prolongation operator aimed at matching the near null-space of the matrix. A specific form of this approach, dubbed "Exact Interpolation Scheme" (EIS), was recently suggested in [5]. In EIS, the solution itself is approximated on the coarse grid, rather than the error as in the classical multigrid. This requires the prolongation operator to be consistently improved as the iterations progress until, ultimately, an accurate solution is obtained by prolongating a coarse approximation. This adaptive approach is also known as the *multiplicative correction scheme.* In this paper we adopt the name EIS but acknowledge that the differences between the various algorithms of this type are very small.

Notable recent developments are Adaptive AMG (αAMG) [7] and adaptive SA (αSA) [6] which were introduced as more general and robust solvers for symmetric linear systems. Here, the multigrid components are built in a separate multilevel setup-phase similar to EIS. The linear system is then solved using the same multigrid

[*]Research supported by the Israeli Science Foundation, grant number 795/08

[†]Department of Computer Science, Technion—Israel Institute of Technology, Haifa, Israel. (eran@cs.technion.ac.il, irad@cs.technion.ac.il.).

components in a separate solution phase. αSA was further developed and adapted to nonsymmetric problems [8].

Adaptive multigrid algorithms have also been used as iterative solvers for computing the principal eigenvector of stochastic matrices (Markov chains). This problem has drawn recent attention, largely due to its relevance in web search applications, among many others. The classical multilevel Markov chain solver of [21] is related to adaptive multigrid. In fact, it is an example of an EIS algorithm, even though it was developed independently. Based on this approach, a collection of recent Markov Chain solvers was suggested in [14, 15, 16, 17, 18, 34]. Appealing as these methods may be, they all suffer from one fundamental common drawback—they calculate the whole multigrid hierarchy of operators in every cycle, with a computational cost that is at best comparable to the cost of the classical AMG setup phase. Experience has shown that, in single processor environments, the setup costs of AMG (or an exact interpolation V-cycle) are typically comparable to those of several classical V-cycles [9, 33]. In [34] it is noted that, empirically, more than 50% of the computation time of each V-cycle is spent on coarse matrix construction. In parallel (distributed memory) environments, the setup phase may require significant communication between nodes and therefore a larger fraction of the total computing time [19].

The approach presented in this paper can be applied with various adaptive multilevel Markov chain solvers. Here, we apply it to a generalized and improved version of the square & stretch (S&S) adaptive multigrid algorithm of [34], while in [30] it is also applied to the AMG-type method of [15] and the pure aggregation method of [21]. The present contribution addresses the aforementioned drawback of adaptive Markov chain solvers—the use of the multiplicative correction approach, which requires us to calculate the multigrid hierarchy in every cycle. We follow the adaptive framework of [6, 7, 8] but incorporate a new scheme that interleaves the classical and EIS algorithms. Like other adaptive approaches, the new scheme uses a solution phase and updates the hierarchy of operators in a setup phase. The new feature is that the update of the hierarchy is done on-the-fly, interlaced with the solution process. We also suggest a parallel variant of this approach. An alternative adaptive approach for Markov chains is presented in [1], where standard solution cycles are also used followed by a more sophisticated setup process.

We next provide the Markov chain problem definition, together with some notation, followed by a brief outline of classical AMG and EIS. Although this work focuses on homogenous systems, the classical techniques were designed as linear system solvers, and therefore will be introduced as such. On the other hand, EIS techniques usually target homogeneous systems and are therefore introduced in that scope.

1.1. Definitions and notation. The following definitions and notation are used throughout this paper.

Spectral radius: $\rho(B)$ denotes the spectral radius of the matrix B. That is, $\rho(B) = \max_i\{|\lambda_i(B)|\}$, where $\lambda_i(B)$ are the eigenvalues of B.

Singular M-Matrix: A matrix A is called an M-Matrix if $A = \mu I - B$, where B is a non-negative matrix and μ is a positive scalar that satisfies $\mu \geq \rho(B)$. If $\mu = \rho(B)$, then A is a *singular M-Matrix*

1.2. Markov chains—problem definition. Let $B \in \mathbb{R}^{n\times n}$ be an 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 [20], there exists a unique vector $\mathbf{x}$ with strictly positive entries that satisfies $B\mathbf{x} = \mathbf{x}$ and $\|\mathbf{x}\|_1 = 1$. 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*

$$A = I - B. \tag{1.1}$$

Our objective is to compute the principal eigenvector of B, i.e., the unique vector $\mathbf{x}$ that satisfies $B\mathbf{x} = \mathbf{x}$, so we seek the solution of the homogeneous system

$$A\mathbf{x} = 0. \tag{1.2}$$

Since B is column-stochastic, $1^T B = 1^T$, where 1 is a constant vector of size n. By (1.1), we have $1^T A = 0$, which means that in this Markov Chain problem the constant vector is the (only) left null-vector of the matrix A.

1.3. Concepts of classical algebraic multigrid. Classical multigrid linear system solvers generally follow a common basic idea. Given the linear system

$$A\mathbf{x} = \mathbf{b}, \tag{1.3}$$

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

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \omega Q^{-1}(A\mathbf{x}^k - \mathbf{b}), \tag{1.4}$$

where ω is a scalar parameter and the matrix Q is an easily inverted simple preconditioner. For example, damped-Jacobi relaxation uses the diagonal of A, $Q = \mathrm{diag}(A)$, and $0 < \omega < 1$. The slow convergence of these relaxations is usually due to only a relatively small number of components in the error, dubbed *algebraically smooth*, that approximately satisfy $Q^{-1}A\mathbf{e} = 0$. For a simple Q, in most cases, these also approximately satisfy the homogenous system (1.2). For this reason, algebraically smooth components are commonly said to be in 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 here by the subscript c). Thus, CGC and relaxation fulfill complementary roles. Algorithm 1 describes a typical two-level classical cycle.

A *multi-level* V-cycle is obtained by recursively treating the coarse-grid problem in step 4. For the coarse-grid operator, the Galerkin or Petrov-Galerkin scheme is usually employed:

$$A_c = RAP, \tag{1.5}$$

where P and R, the prolongation and restriction operators, are chosen to be of full rank. It is well-known and easy to show that if the error before the CGC is in the range of the prolongation and A is non-singular, CGC eliminates it. In this paper the V-cycle is used as a homogenous system solver, with $\mathbf{b} = 0$. Of course, on coarser levels the system

$$A_c\mathbf{e}_c = R\mathbf{r} = \mathbf{r}_c \tag{1.6}$$

is usually inhomogeneous, with $\mathbf{r}_c \neq 0$.

```
Input: Initial vector: $\mathbf{x} \in \mathbb{R}^n$, Right-hand-side vector: $\mathbf{b} \in \mathbb{R}^n$;
  Operators: $A \in \mathbb{R}^{n\times n}$, $P \in \mathbb{R}^{n\times n_c}$, $R \in \mathbb{R}^{n_c\times n}$, $A_c \in \mathbb{R}^{n_c\times n_c}$.
Output: Better approximation for the solution of $A\mathbf{x} = \mathbf{b}$
Algorithm:
 1. Apply pre-relaxations: $\mathbf{x} \leftarrow Relax(A, \mathbf{x}, \mathbf{b})$.
 2. Define the residual $\mathbf{r} = \mathbf{b} - A\mathbf{x}$.
 3. Restrict the residual: $\mathbf{r}_c = R\mathbf{r}$.
 4. Define $\mathbf{e}_c$ as the solution of the coarse-grid problem $A_c\mathbf{e}_c = \mathbf{r}_c$.
 5. Prolong $\mathbf{e}_c$ and apply CGC: $\mathbf{x} \leftarrow \mathbf{x} + P\mathbf{e}_c$.
 6. Apply post-relaxations: $\mathbf{x} \leftarrow Relax(A, \mathbf{x}, \mathbf{b})$.
```

Algorithm 1: Two-level classical cycle

1.4. Concepts of EIS. As noted, the EIS technique can be used either as a setup for the classical approach or as a solver for homogenous systems (1.2), where A is singular.

Suppose that we could construct some prolongation operator P, such that the solution $\mathbf{x}$ of (1.2) 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, substituting $P\mathbf{x}_c$ for $\mathbf{x}$ in equation (1.2), and multiplying through by R, we obtain,

$$RAP\mathbf{x}_c \equiv A_c\mathbf{x}_c = 0\,. \tag{1.7}$$

After solving (1.7), we obtain the sought solution by prolongation: $\mathbf{x} = P\mathbf{x}_c$. This motivates the EIS approach, where the prolongation P is constructed such that the current approximation to the solution is approximately in its range. As the solution becomes more and more accurate, so does the coarse representation of the original problem. The EIS cycle for homogeneous systems is described in Algorithm 2. As before, a multigrid V-cycle is obtained by treating the coarse-grid problem in step 5 recursively.

```
Input: Initial vector: $\mathbf{x} \in \mathbb{R}^n$, fine-grid operator: $A \in \mathbb{R}^{n\times n}$.
Output: Better approximation for the solution of $A\mathbf{x} = 0, ||\mathbf{x}|| = 1$.
  Coarse-grid and transfer operators: $P \in \mathbb{R}^{n\times n_c}$, $R \in \mathbb{R}^{n_c\times n}$, $A_c \in \mathbb{R}^{n_c\times n_c}$.
Algorithm:
 1. Apply pre-relaxations: $\mathbf{x} \leftarrow Relax(A, \mathbf{x})$.
 2. Construct the interpolation operator $P$ with $\mathbf{x}$ approximately 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. Prolong solution: $\mathbf{x} \leftarrow P\mathbf{x}_c$.
 7. Apply post-relaxations: $\mathbf{x} \leftarrow Relax(A, \mathbf{x})$.
```

Algorithm 2: Two-level EIS cycle

When A is nonsymmetric, the restriction needs to be chosen such that the near null-space of A^T is in the range of R^T [8]. In Markov chains, the left null-space is known to be the constant vector, and so we define R with the constant vector in the range of R^T, i.e., $\mathbf{1}_c^T R = \mathbf{1}$. By (1.5), the constant vector is then a left null-vector of A_c.

1.5. The relation between the classical and EIS algorithms. Suppose that we solve the same homogenous problem (1.2) using the two approaches described above. It is interesting to study the relation between the two algorithms. We next show that a two-level version of the EIS approach is actually equivalent to the classical approach, supplemented by adaptive updates of the prolongation and restriction. In the classical approach of Algorithm 1, the coarse-grid problem for $\mathbf{b} = 0$ is given by

$$RAP\mathbf{e}_c = R\mathbf{r} = -RA\mathbf{x}, \tag{1.8}$$

and the CGC is $\mathbf{x} \leftarrow \mathbf{x} + P\mathbf{e}_c$. Suppose that the approximation $\mathbf{x}$ before the CGC is in the range of P, so there exists a vector $\mathbf{x}_c$ that satisfies $\mathbf{x} = P\mathbf{x}_c$. Adding the term $RAP\mathbf{x}_c$ to both sides of (1.8), we get

$$RAP(\mathbf{x}_c + \mathbf{e}_c) = RA(P\mathbf{x}_c - \mathbf{x}) = 0. \tag{1.9}$$

Now, by plugging the vector $\mathbf{w}_c = \mathbf{x}_c + \mathbf{e}_c$ in the left-hand-side of (1.9), we get $A_c\mathbf{w}_c = 0$, which is exactly the same coarse-grid problem as in (1.7). Moreover, by applying the multiplicative CGC (step 6 of Algorithm 2), we get

$$\mathbf{x} \leftarrow P\mathbf{w}_c = P(\mathbf{x}_c + \mathbf{e}_c) = \mathbf{x} + P\mathbf{e}_c,$$

which is exactly the same correction as in the classical approach. Thus, the classical two-level approach is equivalent to EIS, provided that we use the P and R of EIS for both algorithms. This analysis can be readily extended to the multi-level case, provided that P and R are adapted at every level.

The EIS cycle is clearly much more expensive than the classical cycle with 'frozen' operators, because EIS updates P and A_c before each CGC, whereas they remain constant in the classical cycle. The equivalence seen here motivates our approach of interleaving EIS cycles and classical cycles.

1.6. Exact solution on the coarsest grid. The solution on the coarsest grid in the EIS cycles is obtained by calculating the null-vector by an eigensolver or by the shifted inverse power method. The situation in classical cycles, however, requires some care. The matrix A_c is singular on the coarsest grid (see remarks at the end of section 1.4). A solution $\mathbf{e}_c$ may contain an arbitrary proportion of the null-space of A_c, which P may translate to an approximate equivalent null space of the matrix in the upper level. Such a "correction" may be counter-productive, chiselling away at the very null-vector we are trying to expose. Hence, in order to achieve a "null-vector free" approximation on the coarsest grid, we use the singular value decomposition (SVD) of the coarse-grid matrix,

$$A_c = U\Sigma V^T, \tag{1.10}$$

where U and V are orthogonal and Σ is a diagonal non-negative matrix that has the (sorted) singular values of A_c on its diagonal. We define a diagonal matrix Σ^+ whose diagonal is given by

$$\Sigma^+_{ii} = \begin{cases} \Sigma^{-1}_{ii} & \Sigma_{ii} \geq \epsilon \\ 0 & \Sigma_{ii} < \epsilon \end{cases}, \tag{1.11}$$

where ϵ is small and positive (we use $10^{-14} \cdot \|A_c\|$). Note that $\Sigma_{nn} = 0$ because A_c is singular. The coarsest-grid solution is then $\mathbf{x}_c = V\Sigma^+U^T\mathbf{r}_c$, and is a "null-vector free" solution of (1.6). This does not quite guarantee a null-vector free correction $P\mathbf{e}_c$, but the proportion of the null-vector in the correction is relatively small; see also related discussion in [39].

2. Adaptive algebraic multigrid.

For the classical cycle (Algorithm 1), we must choose P such that the algebraically smooth components are approximately in its range. When those are unavailable, adaptive algorithms are usually required. Such algorithms, as in [6, 7], often feature a classical solution phase, preceded by an EIS-like setup phase where the multigrid operators are constructed. The setup phase is targeted at solving the homogeneous system (1.2), aiming to find an approximation (called prototype) for the near null-space of every coarse matrix at each level, and building 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 solution phase using Algorithm 1.

Generally, a more expensive setup is expected to provide a faster solution process; however, it is hard to optimize the overall cost of the setup and solution processes. In addition, there might be problems where a fixed setup may not suffice for an effective solution cycle. In [6, 7], the initial setup cycle is followed by a rather expensive self-testing procedure. This involves applying several solution cycles to a random guess and monitoring the convergence rate of the solver. If it is too poor, then more setup cycles are performed. Setup cycles with more pre and post relaxations and with standard cycles as relaxations were also considered. In that work, however, the algorithm aims at solving multiple systems of the same matrix with different right hand sides. Therefore, this effort is clearly worthy, because a single setup is used for solving many linear systems. Also, [6] aims at developing solvers for systems with a rich near null-space so more effort is invested in the setup. Here, we aim only to find the unique null-vector of the matrix, so this procedure might be overly expensive in our case. We therefore exploit the fact that the EIS and classical algorithms target the same homogenous problem, so they can enhance each other.

We next describe three approaches for combining setup and solution cycles. All approaches have four main steps and differ in only one or two of those steps. These approaches will be examined and compared later in this paper.

2.1. A simple adaptive AMG scheme.

The first approach is to initially perform several setup cycles until the operators reach a certain quality and then continue and apply classical solution cycles. We denote this approach by AFTER(ε_α), and we assume that the quality of the operators is mostly controlled by the smoothness of the prototype $\mathbf{x}$ which is used for defining the prolongation P. More specifically, we perform several relaxation sweeps on an initial guess, followed by an initial setup cycle. Then we perform EIS cycles until the residual norm of the approximation drops below ε_α. Finally, we perform one additional EIS cycle so that the resulting operators will be based on the approximate solution satisfying this criterion. For the rest of the process we use standard solution cycles. The algorithm is described in Algorithm 3, where $V_{SOL}(\cdot)$ denotes a solution cycle (Algorithm 1 applied with $\mathbf{b} = 0$), and $V_{EIS}(\cdot)$ denotes a setup cycle which also creates/updates the MG operators (Algorithm 2). This notation is used for the rest of this paper.

A rule of thumb for ε_α is that it should be smaller than the magnitude of the smallest non-zero eigenvalue in absolute value. For the test cases and solution method considered in this work, $\varepsilon_\alpha = 10^{-5}$ is sufficient.

```
Input: Threshold $\varepsilon_\alpha$, fine-grid operator: $A \in \mathbb{R}^{n\times n}$, initial guess $\mathbf{x}_0$.
Output: Approximate solution to: $A\mathbf{x} = 0, ||\mathbf{x}|| = 1$
Algorithm: Perform setup followed by solution cycles
 1. Initial Setup:
    Apply a few relaxations to smooth $\mathbf{x}_0$.
    Do an initial EIS cycle: $\mathbf{x} \leftarrow V_{EIS}(\mathbf{x}_0)$.
    % $\mathbf{x}_0$ after pre-relaxations is in the range of $P$.
    if $||A\mathbf{x}_0||_1 < \varepsilon_\alpha$ goto Step 4.
 2. Improve solution approximation:
    while $||A\mathbf{x}||_1 > \varepsilon_\alpha$ do $\mathbf{x} \leftarrow V_{EIS}(\mathbf{x})$.
 3. Finalize setup:
    % Final $P$ based on pre-relaxed $\mathbf{x}$.
    $\mathbf{x} \leftarrow V_{EIS}(\mathbf{x})$.
 4. Solution:
    Apply $\mathbf{x} \leftarrow V_{SOL}(\mathbf{x})$ until convergence.
```

Algorithm 3: AFTER—a simple adaptive AMG scheme

2.2. On-the-fly adaptive algebraic multigrid. In the"on-the-fly" approach we first apply a single initial setup cycle and start the solution process. Additional setup cycles may be applied as the solution phase progresses. The underlying assumption is that solution cycles are considerably cheaper, but they require the operators supplied by the EIS setup. Again, we assume that better approximations supplied to the EIS cycle yield better operators in return.

The goal of the on-the-fly algorithm is to reach the accuracy $||A\mathbf{x}||_1 < \varepsilon_\alpha$ as fast as possible, preferring solution cycles over EIS cycles. We try to save computations by using the operators from previous setup cycles. Even though these operators may not be very accurate, they may suffice for obtaining useful solution cycles. For example, assuming that an EIS cycle costs like five solution cycles and achieves a convergence factor of about 0.4 (which is quite reasonable for difficult problems), a solution cycle that achieves an uninspiring convergence factor of about 0.75 may be more efficient, because $0.75^5 < 0.4$. With this motivation, we introduce the following procedure where $q(\cdot)$ is a quality measure of the approximation, and $0 < \gamma \leq 1$ is a scalar threshold for an acceptable convergence factor of the solution cycles.

Procedure try-SOL-else-EIS(γ):

```
1. $\mathbf{y} = V_{SOL}(\mathbf{x})$.
2. if $q(\mathbf{y}) > q(\mathbf{x})$ do $\mathbf{x} \leftarrow V_{EIS}(\mathbf{x})$ and finish.
3. if $q(\mathbf{y}) < \gamma q(\mathbf{x})$ then $\mathbf{x} = \mathbf{y}$, else $\mathbf{x} \leftarrow V_{EIS}(\mathbf{y})$.
```

The choice of $q(\mathbf{x})$ is not obvious. If A is symmetric semi-definite then the Rayleigh quotient is a useful measure:

$$q(\mathbf{x}) = \frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T\mathbf{x}}. \tag{2.1}$$

However, for nonsymmetric problems, which are the focus of this paper, this measure is inappropriate and we use instead

$$q(\mathbf{x}) = \frac{\|A\mathbf{x}\|_1}{\|\mathbf{x}\|_1}\,, \tag{2.2}$$

which means that we measure the convergence factor by the l_1 residual norm. The residual, which emphasizes high-energy modes, may not generally be an optimal measure of performance. However, we find it quite reliable in practice when used, as here, for choosing between two rather similar multigrid processes. With this procedure, the (non-overlapping) on-the-fly algorithm proceeds as in Algorithm 4.

```
Input: Threshold $\varepsilon_\alpha$, Convergence parameter $\gamma$, operator: $A \in \mathbb{R}^{n\times n}$, initial
        guess $\mathbf{x}_0$.
Output: Approximate solution to: $A\mathbf{x} = 0, ||\mathbf{x}|| = 1$
Algorithm: Solution by non-overlapping setup and solution cycles
 1. Initial Setup:
    Apply a few relaxations to smooth $\mathbf{x}_0$.
    Do an initial EIS cycle: $\mathbf{x} \leftarrow V_{EIS}(\mathbf{x}_0)$.
    % $\mathbf{x}_0$ after pre-relaxations is in the range of $P$.
    if $||A\mathbf{x}_0||_1 < \varepsilon_\alpha$ goto Step 4.
 2. Improve Solution Approximation:
    while $||A\mathbf{x}||_1 > \varepsilon_\alpha$ do try-SOL-else-EIS($\gamma$).
 3. Finalize Setup:
    % Final $P$ based on pre-relaxed $\mathbf{x}$.
    $\mathbf{x} \leftarrow V_{EIS}(\mathbf{x})$.
 4. Solution:
    Apply $\mathbf{x} \leftarrow V_{SOL}(\mathbf{x})$ until convergence.
```

Algorithm 4: On-the-fly adaptive AMG

2.3. Overlapping on-the-fly adaptive algebraic multigrid. The last strategy we consider is mostly suitable for distributed-memory parallel computations. Publications on adaptive multigrid have not focused on parallel computation. However, since classical and adaptive AMG have so many common ingredients, the parallelization of adaptive AMG should probably be very similar to that of classical AMG. With this in mind, we immediately note a potential drawback. In serial code, the setup cycle, which is similar to EIS, typically costs like several (less than 10) solution cycles [9, 33, 19, 11]. In parallel, however, the setup requires much shorter and more frequent communication between computing nodes. Coarsening procedures, matrix multiplication, construction of P and R, and other ingredients of the setup cycle, are typically much more complicated than the simple matrix-vector multiplications of the solution cycle. Numerical results in [40, 31] show a ratio of about 15-45, and in many cases the setup cost exceeds the total solution cost. In addition, the algorithms are shown to be scalable with the number of processors only when the problem size is growing as well (weak scaling). That is, each processor handles a fixed amount of grid points, and as the problem grows more processors are used. If we increase the number of processors for a fixed problem size, and refine the grid-point partitioning between processors, then we impose more communication between processors and hence a higher cost. This means that for a given problem size there is an optimal amount of processors, and exceeding it will harm the performance. Also, in [31], one clearly sees that the ratio between the setup cost and iteration cost grows with the number of processors (or, equivalently, with the problem size). As a result, in parallel adaptive AMG computations we should limit the amount of setup cycles performed even more than in serial computations.

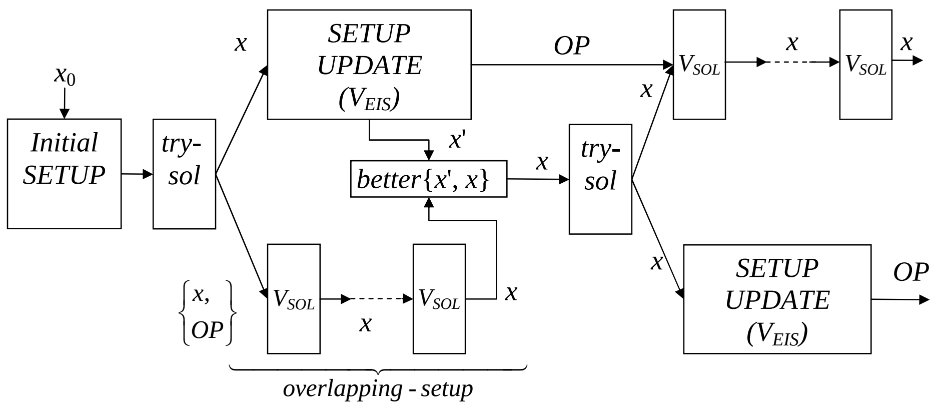

FIG. 2.1. ***Overlapping on-the-fly adaptive AMG:*** *A scheme for combining setup improvements in parallel to classical solution cycles for the solution of Markov chains. Solution cycles are denoted by "V_{SOL}", and "OP" denotes the hierarchy of multigrid operators.*

The above discussion highlights even more the advantage of Algorithm 4 over Algorithm 3 in parallel computations. However, the overlapping on-the-fly algorithm that we propose next aims at improving Algorithm 4 even further by using double computing resources. The new algorithm uses high-level parallelism between two computing clusters that does not require frequent communication between them. One cluster applies solution cycles while the other performs a setup update. It is important to note that if doubling the resources for an internally parallelized Algorithm 4 significantly improves its runtime, then it might be better than the overlapping approach we propose (ideally, the runtime would drop by a factor of 2). However, if adding more computing resources does not improve the performance of Algorithm 4 for a given problem size, then the overlapping version is worthy of consideration.

```
Input: Threshold $\varepsilon_\alpha$, Convergence parameter $\gamma$, operator: $A \in \mathbb{R}^{n\times n}$.
Output: Approximate solution to: $A\mathbf{x} = 0, ||\mathbf{x}|| = 1$
Algorithm: Solve by overlapping setup and solution cycles
 1. Initial Setup:
    Apply a few relaxations to smooth $\mathbf{x}_0$.
    Do an initial EIS cycle: $\mathbf{x} \leftarrow V_{EIS}(\mathbf{x}_0)$.
    % $\mathbf{x}_0$ after pre-relaxations is in the range of $P$.
    if $||A\mathbf{x}_0||_1 < \varepsilon_\alpha$ goto Step 4.
 2. Improve Solution Approximation:
    while $||A\mathbf{x}||_1 > \varepsilon_\alpha$ do
    % The next two lines are marked by the 'try-sol' block on Fig. 2.1.
    • $\mathbf{y} \leftarrow V_{SOL}(\mathbf{x})$.
    • if $q(\mathbf{y}) < \gamma q(\mathbf{x})$ then $\mathbf{x} \leftarrow \mathbf{y}$;
      else $\mathbf{x} \leftarrow$ overlapping-setup($better(\mathbf{x}, \mathbf{y})$).
 3. Finalize Setup:
    $\mathbf{x} \leftarrow$ overlapping-setup($\mathbf{x}$)
 4. Solution:
    Apply solution cycles until convergence.
```
Algorithm 5: Overlapping on-the-fly adaptive AMG

The overlapping on-the-fly adaptive algorithm is exhibited in a block diagram in Figure 2.1. The first EIS cycle is an essential setup phase. Once an initial hierarchy of operators is set, the solution process is good to go. As in the non-overlapping version in Algorithm 4, we apply solution cycles so long as they are efficient (marked by the 'try-sol' block), until either their performance deteriorates or the condition $||A\mathbf{x}||_1 < \varepsilon_\alpha$ is satisfied. Then, the hierarchy update evolves via EIS in its slow pace while the solution cycles are applied in parallel. Once a new hierarchy is computed, a cross-over of information may occur, whereby the solution process provides the setup process with a more accurate approximation, while the setup process provides the solution process with a new hierarchy of operators. At each cross-over, the two computing units switch roles of applying solution and setup phases, so as not to require transferring of the operators (a large amount of data) between the computing nodes. Since the two processes join only at cross-over points, this scheme does not require recurrent and frequent communication between the setup and solution clusters.

The function *better* that appears in the cross-over in Figure 2.1 is optional. Because there is no guarantee that a single setup cycle yields a convergent solution process, we may use this safety procedure that aims to limit unnecessary computations. In practice, one may use

$$\mathit{better}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \quad \text{if} \quad q(\mathbf{x}_1) < q(\mathbf{x}_2); \quad \text{else} \;\; \mathbf{x}_2,$$

where $q(\mathbf{x})$ is the quality measure mentioned earlier in Equations (2.1)-(2.2). We note that at least in the experiments considered in this paper, the vector $\mathbf{x}'$ from the setup cycle is almost never preferred by the function 'better'.

The overlapping on-the-fly adaptive MG algorithm appears in Algorithm 5, where the following procedure is used for overlapping solution and setup cycles:

Procedure overlapping-setup(x):

- Do setup update on cluster A:
  $\mathbf{x}' \leftarrow V_{EIS}(\mathbf{x})$.
- Do solution cycles on cluster B:
  repeat $\mathbf{x} \leftarrow V_{SOL}(\mathbf{x})$ **until** cluster A finishes or iterations diverge.
- $\mathbf{x} \leftarrow \mathit{better}(\mathbf{x}, \mathbf{x}')$

2.4. A qualitative analysis. To provide some insight into the properties of the different strategies, we resort to a simplified qualitative analysis. Given current multigrid operators and an approximation $\mathbf{x}$, we consider three scenarios:

1. Applying solution cycles with the existing operators until convergence;
2. Applying an EIS cycle and then solution cycles until convergence;
3. Applying r solution cycles, followed by an EIS cycle with the new approximation, and then solution cycles until convergence;

We examine the total time of solution for a fixed problem size in the three scenarios. We measure the approximate time for convergence in work units, each equal to the cost of one solution cycle. The convergence criterion is an error reduction by a factor $\epsilon \ll 1$. For this discussion we use the notation given in Table 2.1. We expect, and indeed typically observe, $0 < \gamma_E < \gamma_2 < \gamma_1 < \gamma_0 \leq 1$.

2.4.1. Serial environment. Table 2.2 summarizes the costs for the total solution processes, where n denotes the number of solution cycles until convergence after the operators are finalized. To select the preferred strategy we need to know all the quantities in the equations, and, since those are problem and solver dependent, it is hard to choose the best option in advance. For example, in the unlikely case that EIS

Symbol	Description
γ_0	The average convergence factor when applying solution cycles with the existing operators, as in scenario 1.
γ_E	The convergence factor of an EIS cycle. For simplicity, we assume that it does not depend on the input $\mathbf{x}$.
R	The ratio between the computational costs of EIS and a solution cycle. An EIS cycle costs the same as R solution cycles.
γ_1	The average convergence factor of the solution cycles using operators after an additional EIS cycle, as in scenario 2.
γ_2	The average conv. factor of solution cycles using operators produced by an EIS cycle that followed r solution cycles, as in scenario 3.

TABLE 2.1
Qualitative analysis notation

Scenario	Derivation	Approximate Cost
1	$\gamma_0^n = \epsilon$	$\dfrac{\log \epsilon}{\log \gamma_0}$
2	$\gamma_E \gamma_1^n = \epsilon$	$R + \dfrac{\log(\epsilon/\gamma_E)}{\log \gamma_1}$
3	$\gamma_0^r \gamma_E \gamma_2^n = \epsilon$	$r + R + \dfrac{\log(\epsilon/\gamma_0^r \gamma_E)}{\log \gamma_2}$

TABLE 2.2
Qualitative analysis: solution costs

does not improve the performance of the solver significantly, i.e., $\gamma_0 \approx \gamma_1 \approx \gamma_2$, then the simple first strategy is best. On the other hand, if the EIS cycle is equally effective regardless of the stage in which we carry it out, i.e., $\gamma_0 >> \gamma_1 \approx \gamma_2$, then, unless R is extremely large, the second strategy will be best. In this case, the third strategy is second-best because the initial r cycles of factor γ_0 are relatively less effective.

In this work we focus on hard problems, that require high-quality MG operators. In such cases we typically find $\gamma_2 << \gamma_1 << \gamma_0$ or $\gamma_2 << \gamma_1 \approx \gamma_0$. Then, the third strategy, which corresponds to the on-the-fly approach, will have the best asymptotic convergence factor, and will likely have the best performance.

2.4.2. Parallel environment. In scenarios 2 and 3, we also consider the case where solution cycles are applied in parallel to the setup cycle. As mentioned before, we double the resources by engaging a second computing cluster but assume that doubling the resources in one cluster does not lead to significant speedup. Table 2.3 summarizes the solution process costs of both scenarios using overlapping setup and solution cycles.

¿From this analysis one can see that the overlapping scheme best suits situations where the EIS cycles are very expensive compared to the solution cycles, i.e., R is rather big as is indeed expected in parallel-AMG computations. If γ_0 is not very close to 1, then the solution cycles run in parallel to the EIS cycle are still effective. Then, the overlapping approach gains from the R solution cycles, and in addition we obtain good asymptotic behavior. This approach is also good for $\gamma_0 \approx \gamma_1 \approx \gamma_2$, where the first strategy is best, though it requires more computing units. In the non-overlapping approaches (Table 2.2), an excessively large R might render the second and third scenarios ineffective due to the large cost of EIS. However, if R is moderately

Scenario	Derivation	Approximate Cost
2	$\min\{\gamma_E, \gamma_0^R\}\gamma_1^n = \epsilon$	$R + \dfrac{\log(\epsilon/\min\{\gamma_E, \gamma_0^R\})}{\log\gamma_1}$
3	$\gamma_0^r \min\{\gamma_E, \gamma_0^R\}\gamma_2^n = \epsilon$	$r + R + \dfrac{\log(\epsilon/\gamma_0^r \min\{\gamma_E, \gamma_0^R\})}{\log\gamma_2}$

TABLE 2.3
Qualitative analysis: solution costs in parallel environment using overlapping setup

high, and γ_0 is not sufficiently small, the second and third strategies come out ahead. In any case, the overlapping strategy is optimal in both situations, and therefore it is very useful if R is big and the additional hardware cost is acceptable.

3. Adaptive Petrov-Galerkin smoothed aggregation for Markov chains.

In this section we describe the method of Petrov-Galerkin smoothed aggregation that we use together with the on-the-fly approach for the solution of Markov chains. This method is related to the Square & Stretch algorithm of [34].

3.1. Tentative aggregation/disaggregation operators. Recall that A is a singular M-Matrix and the solution is positive. Assume we have a partitioning of the variables in $\mathcal{N} = \{1, ..., n\}$ into disjoint subsets $\{\mathcal{C}_J\}_{J=1}^{n_c}$ called aggregates (their construction appears later). Using these aggregates we first define tentative prolongation and restriction P_0 and R_0 that are simple aggregation/disaggregation operators. Also, let $\mathbf{x}$ be a positive approximate solution for (1.2).

Given the aggregates $\{\mathcal{C}_J\}_{J=1}^{n_c}$ and the approximation $\mathbf{x}$, we define a tentative aggregation-based prolongation matrix P_0 with $\mathbf{x}$ in its range. As noted earlier, the left null space of the matrix A is the constant vector. Therefore, we choose all restriction matrices to be piece-wise constant, so that the matrices on all the levels have a constant left null-vector. Specifically, given $\mathbf{x}$ and the aggregates $\{\mathcal{C}_J\}_{J=1}^{n_c}$, we use the matrices originally suggested in [21]

$$R_{J,i} = \begin{cases} 1 & \text{if } i \in \mathcal{C}_J \\ 0 & \text{otherwise} \end{cases} \qquad P_{i,J} = \begin{cases} \mathbf{x}_i/\left(\mathbf{x}_c\right)_J & \text{if } i \in \mathcal{C}_J \\ 0 & \text{otherwise} \end{cases}, \tag{3.1}$$

with $\left(\mathbf{x}_c\right)_J = \frac{1}{|\mathcal{C}_J|}\sum\limits_{r\in\mathcal{C}_J} \mathbf{x}_r$, where $|\mathcal{C}_J|$ denotes the number of nodes in the aggregate $\mathcal{C}_J$. Note that there is a single non-zero value in each row of P_0 and R_0^T, and their sparsity structure is identical.

3.2. Smoothing the tentative operators. In order to improve the tentative operators, fixed and adaptive SA techniques were introduced for symmetric problems [6, 35, 36, 38], and Petrov-Galerkin SA [8, 12, 13, 29, 37] for nonsymmetric problems, including Markov chains [14]. The main concept of smoothed aggregation is that the transfer operators are smoothed with a simple smoothing operator, generally of the form

$$S_\omega(A) = I - \omega Q^{-1}A, \tag{3.2}$$

where Q is a preconditioner of A and ω a positive scalar parameter, usually a damping parameter. These are similar, though not necessarily identical, to the preconditioner and damping parameter employed in the relaxation. The prolongation is thus given by

$$P_\omega = S_\omega(A)P_0. \tag{3.3}$$

For symmetric problems, the standard SA algorithm uses $R_\omega^T = P_\omega$. For nonsymmetric problems, a Petrov-Galerkin approach is often applied, whereby $R^T \neq P$. A reasonable choice for smoothing the restriction may be [14]

$$R_\omega^T = S_\omega(A^T)R_0^T. \tag{3.4}$$

The ω's for R and P may differ [29].

In this work, we focus on the case where the prolongation is smoothed as in (3.3), but the restriction remains R_0. In this case, the prolongation smoother in (3.2) is usually chosen to be [12, 13, 29, 37]

$$S_{\rho^{-1}}(A) = I - \frac{1}{\rho(Q^{-1}A)}Q^{-1}A. \tag{3.5}$$

The spectral radius is usually approximated using an Arnoldi-type method, and this works well for symmetric matrices. Since the eigenvectors of the matrix are orthogonal in the symmetric case, the Rayleigh Quotient as in (2.1) is guaranteed to be bounded by the spectral radius. However, finding the spectral radius of a nonsymmetric matrix may be costly, because the simple Rayleigh Quotient may provide a value far larger then the spectral radius. Nevertheless, we do estimate the spectral radii, but only once in the first setup cycle and with low accuracy. Specifically, in this work we apply 25 power-method iterations and use the Rayleigh Quotient (2.1) as an approximation to the spectral radius. This calculation is relatively expensive, and alternatives are worthy of investigation. Simpler options such as using constant smoothing parameters or using matrix norms (upper bound to the radii) were found to be significantly less effective than using the true spectral radius.

In [13], an overcorrection technique was used to improve the performance of smoothed aggregation. Similarly, we apply over-correction,

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha P\mathbf{e}_c\,, \tag{3.6}$$

instead of the regular CGC in Step 5 of Algorithm 1, where $\alpha > 1$ is the over-correction parameter. Correspondingly,

$$\mathbf{x} \leftarrow (1-\alpha)\mathbf{x} + \alpha P\mathbf{x}_c \tag{3.7}$$

is used in step 6 of Algorithm 2. In all our tests we fix $\alpha = 1.1$.

Finally, in [12, 37] it was advised to perform an additional pre-relaxation with the same operator (3.5) used for smoothing P. Motivated by this, we modify $\mathbf{x}$ after computing the residual that is transferred to the coarser level. That is, we add

$$\mathbf{x} \leftarrow \mathbf{x} + \tfrac{1}{\rho(Q^{-1}A)}Q^{-1}\mathbf{r}\,. \tag{3.8}$$

to step 2 of Algorithm 1, where Q is as in (3.5). Such use of the residual was suggested already in [10]. Note that this modification is very cheap as it does not require a matrix-vector multiplication. Using this procedure, the smoother (3.5) acts on both $\mathbf{x}$ and the correction $P_0\mathbf{e}_c$, instead of $P_0\mathbf{e}_c$ alone. From linearity considerations, the CGC operator of our SA version of Algorithm 1 is then given by

$$S_{\rho^{-1}}(I - \alpha P_0(A_c)^+R_0A) \tag{3.9}$$

instead of

$$(I - \alpha S_{\rho^{-1}}P_0(A_c)^+R_0A), \tag{3.10}$$

as it would be without this modification. This small and cheap change has a significant effect, almost the same as a post-relaxation.

3.3. Defining aggregation. SA with no R smoothing is generally considered inferior to smoothing both R and P [29]. However, the comparisons that led to this conclusion employed the same aggregates for both methods. When smoothing both R and P, the eventual coarse-grid operator in (1.5) has the form of a cubic polynomial in A, coarsened by the tentative operators. Unless the coarsening is aggressive enough, a significant stencil growth might occur on coarser grids. On the other hand, smoothing only P results in a coarsened quadratic polynomial in A, which allows less aggressive coarsening. For a discretization of a d-dimensional second-order PDE on a rectangular grid, for example, smoothing both operators requires aggregates of width 3 in each spatial direction (aggregates of size 3^d), while smoothing only P requires aggregates of width 2 [13]. The use of smaller aggregates is the key to the potential advantage of smoothing only P, since less aggressive coarsening (more coarse-grid DOFs) generally leads to a better coarse representation of the fine-grid problem, and therefore to a better CGC.

```
Input: $s$—typical aggregate size, $W \in \mathbb{R}^{n\times n}$—strong connections matrix
Output: Aggregates: $\{\mathcal{C}_J\}_{J=1}^{n_c}$.
Algorithm:
Let $U = 1,...,n$ be a set of unassigned elements
repeat
    Among the elements in $U$, find the element $i$ with the least remaining
    neighbors in $U$.
    Initialize a new aggregate $\mathcal{C} = \{i\}$.
    if i has more than one neighbor then
        Define $U_{near} \subseteq U$ to contain all elements whose distance from $i$ is at
        most $\lfloor \frac{s}{2} \rfloor$.
        Find all circles within $U_{near}$ of length $s$ or less that contain $i$.
        % At least one circle will always be found.
        Choose the circles of maximal length, and amongst those choose the
        circle of maximal sum of inner connections in $W$.
        Add the members of the chosen circle to $\mathcal{C}$.
    else
        Let $p$ be the only neighbor of $i$ in $U$.
        Add $p$ to $\mathcal{C}$
        if other neighbors of p with only one neighbor besides p exist then
            Add up to $(s - 2)$ of them to $\mathcal{C}$.
        end
    end
    Remove the chosen aggregate's members from $U$.
    if After removing the chosen aggregate there are elements with no
    neighbors then
        Add the elements with no neighbors to $\mathcal{C}$.
    end
until all elements are assigned to aggregates;
```
Algorithm 6: *Bottom-up aggregation technique*

For unstructured grids, the classical neighborhood aggregation of [38] is appropriate in the case where both R and P are smoothed, but it is too aggressive when only P is smoothed [29]. Therefore, we use the *Bottom-Up* method of [34], which aggregates nodes that satisfy a circular dependency, cf., Algorithm 6. The weight matrix W is defined as in [34], based on the elements of the matrix A and the current approximation $\mathbf{x}$ to its near null-space. Specifically, we use a connectivity matrix $\hat{W}$ with

$$\hat{W}_{ij} = \begin{cases} -A_{ij}\mathbf{x}_j & \text{if } i \neq j \text{ and } -A_{ij}\mathbf{x}_j \geq \theta \max_{l \neq i} -A_{il}\mathbf{x}_l\,, \\ 0 & \text{otherwise,} \end{cases} \tag{3.11}$$

where we use $\theta = 0.1$. Then, we symmetrize our connectivity matrix as follows:

$$W = \frac{1}{2}\left(\hat{W} + \hat{W}^T\right)\,. \tag{3.12}$$

Since W is symmetric, each pair of neighbors is a circle of length 2.

For structured grids, this technique yields aggregates of width 2 in each spatial direction in many scenarios we have studied, in 1D, 2D and 3D. Investigating and improving this aggregation method remains the subject of ongoing research. Another efficient alternative aggregation method which is appropriate to smoothing only P appears in [25, 24].

4. Numerical Results.

In this section, we demonstrate and compare between the three algorithms discussed in Section 2 for integrating solution and setup cycles. The algorithms are applied to adaptive Petrov-Galerkin SA, where only the prolongation is smoothed by the Jacobi operator (Q is the diagonal part of the matrix A) with damping parameter as in (3.5). We show results for three Markov chain problems that were found to be difficult enough to require more than one setup cycle.

In all the results shown, we start with an initial random guess and reduce its l_1 residual norm $||A\mathbf{x}||_1$ by a factor of 10^{10}. We employ V(2,1) solution cycles with Jacobi relaxations. Exact solution is performed once the problem size is below 16. Similarly to [6, 7], when we perform setup cycles followed by solution cycles in the AFTER and OnTheFly algorithms, we do more pre-relaxations in the EIS cycles than in the solution cycles: 4 pre-relaxations and 1 post-relaxation. We do that in all setup cycles of OTF and in the last EIS cycle of AFTER (Step 3). All other EIS cycles are V(2,1). Note, that a $V_{EIS}(4,1)$ cycle is not significantly more costly than $V_{EIS}(2,1)$, because most of the computation is spent on operator construction and matrix multiplication (RAP calculations) [6, 7, 34]. The relaxations are damped with

$$\omega = \frac{4}{3\rho(D^{-1}A)}, \tag{4.1}$$

where the value 4/3 minimizes the expression

$$\min_{\omega} \max_{0.5 \leq x \leq 1} |1 - \omega x|,$$

that corresponds to the Jacobi relaxation we are using.

In the following tables and figures we show computation times as well as work units. The latter is defined as the cost of a single $V_{SOL}(2,1)$ solution cycle. The experiments were performed using MATLAB R2010b on a machine with an Intel core i7 CPU with 8 GB of RAM memory, running Linux Ubuntu.

For the first approximation to the null-space on which we base the initial operators, we do 20 relaxation sweeps on the initial random guess, counting these as 3 work units (which assumes that about 7 relaxations cost the same as a V(2,1) solution

size	*lev*	C_{op}	$u_{set}(t_{set})$	*algorithm*	V_{EIS}, V_{SOL}	γ	$u_{sol}(t_{sol})$
65536	7	1.64	23 (0.35s)	EIS	15 , 0	0.34	63 (0.91s)
				AFTER(10^{-2})	0 , 32	0.64	35 (0.44s)
				AFTER(10^{-3})	2 , 16	0.45	27 (0.35s)
				AFTER(10^{-4})	3 , 12	0.36	28 (0.37s)
				AFTER(10^{-5})	5 , 10	0.34	33 (0.44s)
				OnTheFly(10^{-4})	2 , 13	0.34	25 (0.33s)
262144	8	1.65	22 (1.00s)	EIS	16 , 0	0.36	85 (3.97s)
				AFTER(10^{-2})	0 , 61	0.77	64 (2.92s)
				AFTER(10^{-3})	2 , 22	0.54	35 (1.60s)
				AFTER(10^{-4})	3 , 15	0.43	33 (1.50s)
				AFTER(10^{-5})	5 , 11	0.35	39 (1.80s)
				OnTheFly(10^{-5})	2 , 15	0.35	30 (1.44s)

TABLE 4.1
Tandem Queue results

cycle). Then we perform the first setup cycle, which also includes the aggregation construction and estimation of the spectral radii. This cycle is identical in all settings and the aggregates, as well as the spectral radii, are calculated only once in that cycle and kept fixed for the rest of the solution process. u_{set} and t_{set} represent the work-units and time (in seconds) of the first setup cycle, while u_{sol} and t_{sol} are the work-units and timings of the solution without the initial setup cycle. V_{EIS}, V_{SOL} are the number of EIS cycles (excluding the initial one) and solution cycles, respectively. The operator complexity denoted by C_{op} is the total number of non-zeros in the matrix A and all its coarse-grid approximations, divided by the number of non-zeros of the fine-grid A. The operator complexity and number of levels are similar in all settings since all use the same Bottom-Up algorithm with a preferred aggregate size of $s = 4$. The asymptotic convergence factor is denoted by γ and is defined as the geometric mean of the residual drop factor of the last 5 solution cycles.

For the smaller-sized problems with about 65,000 unknowns, we use $\varepsilon_\alpha = 10^{-4}$ as a default safe accuracy threshold for the algorithms considered. For the larger problems of about 262,000 unknowns, we use $\varepsilon_\alpha = 10^{-5}$.

4.1. Serial environment experiments.

We compare the AFTER algorithm with various values of ε_α to the on-the-fly algorithm with the safe pre-chosen ε_α, and a convergence parameter of $\gamma = 0.75$ (detailed in Algorithm 4).

4.1.1. Tandem Queue Markov Chain.

The first problem considered is the tandem queueing network problem appearing in [1, 14, 15, 21, 34]. This problem is nonsymmetric and has a significantly complex spectrum, which lies in the triangle whose vertices are $-0.5 \pm \frac{\sqrt{3}}{2}i$ and 1. The stencil of the matrix A is given by

$$\begin{pmatrix} & & -\mu_1 \\ -\mu & 1 & \\ & -\mu_2 & \end{pmatrix},$$

where $\mu = 10/31$, $\mu_1 = 11/31$ and $\mu_2 = 10/31$.

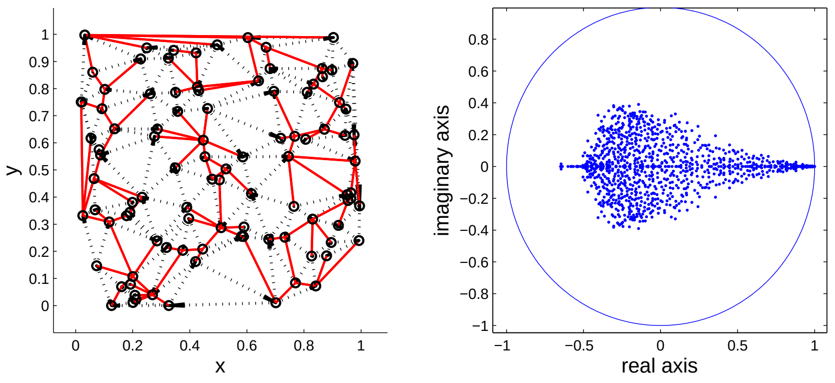

FIG. 4.1. *Left: Nonsymmetric random planar graph—directed edges are marked with dotted lines while undirected edges are marked with a solid line. Right—the spectrum of the Markov chain bounded by the unit circle.*

Table 4.1 shows the results for the Tandem Queue problem using the EIS algorithm as a solver (Algorithm 2), AFTER algorithm (Algorithm 3) with various choices of ε_α, and results for the non-overlapping on-the-fly algorithm (Algorithm 4).

The results show that a high accuracy of the prototype is required for this test-case. More setup cycles in the AFTER algorithm, or equivalently using a smaller ε_α, produced better asymptotic convergence factors. The AFTER and OnTheFly algorithms, with sufficiently small ε_α, achieved asymptotic convergence rates similar to EIS. The cost of the EIS iterations, however, is significantly higher. The timings of AFTER clearly show the tradeoff: more setup cycles gain better asymptotic convergence, but require more computational time. Cost-wise, $\varepsilon_\alpha = 10^{-3}$ or 10^{-4} is the best compromise for this test case, even though the resulting convergence factors are not optimal. The on-the-fly algorithm achieved the best performance, using several solution cycles to reach the required accuracy instead of some of the setup cycles in AFTER.

4.1.2. Random walk on a nonsymmetric random planar graph. The next test problem is a random walk on a nonsymmetric unstructured planar graph demonstrated in Figure 4.1. A similar problem with a symmetric graph appears in [14, 15, 34], and a slightly different nonsymmetric version is considered in [17, 18]. The graph is generated by choosing n random points in a unit square and applying Delaunay triangulation. Then, a spanning tree of the graph is generated by a DFS algorithm and its edges are kept undirected. For the remaining edges, a direction is chosen at random. The spanning tree edges are kept undirected in order to ensure that the graph is strongly connected and hence the associated matrix is irreducible. The weight of an edge (i, j) is determined by the reciprocal of the outgoing degree of node i, that is, we normalize the columns of the binary matrix representing the graph to make it column-stochastic.

Table 4.2 shows the results for this problem. Here, the accuracy required for the prototype vector was lower than in the previous problem. In the case of AFTER, requiring excessive accuracy from the prototype increases the solution time due to the expensive setups. However, such a large default choice of $\varepsilon_\alpha = 10^{-2}$ is too risky in general, and may result in very poor behavior in many other problems. Also, this

size	*lev*	C_{op}	$u_{set}(t_{set})$	*algorithm*	V_{EIS}, V_{SOL}	γ	$u_{sol}(t_{sol})$
65536	7	2.03	20 (0.46s)	EIS	15 , 0	0.31	66 (1.62s)
				AFTER(10^{-2})	0 , 20	0.46	23 (0.42s)
				AFTER(10^{-3})	3 , 11	0.29	29 (0.57s)
				AFTER(10^{-4})	4 , 10	0.28	32 (0.65s)
				AFTER(10^{-5})	6 , 8	0.28	37 (0.82s)
				OnTheFly(10^{-4})	1 , 14	0.28	22 (0.42s)
262144	8	2.04	25 (1.92s)	EIS	16 , 0	0.36	86 (7.79s)
				AFTER(10^{-2})	0 , 25	0.50	28 (1.95s)
				AFTER(10^{-3})	3 , 13	0.38	32 (2.56s)
				AFTER(10^{-4})	4 , 12	0.37	35 (2.91s)
				AFTER(10^{-5})	6 , 10	0.36	43 (3.89s)
				OnTheFly(10^{-5})	1 , 15	0.36	26 (1.91s)

TABLE 4.2
Result Summary: Random walk on a nonsymmetric random planar graph

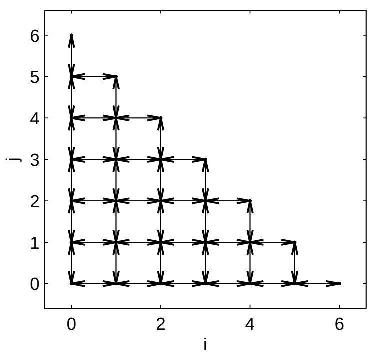

FIG. 4.2. *Triangular lattice of 28 states*

choice has the worst asymptotic convergence ($\gamma \approx 0.5$). The on-the-fly approach does not suffer from this drawback and performs essentially as well as the best choice of AFTER in terms of solution cost, and with a better asymptotic convergence rate ($\gamma \approx 0.36$). The pure EIS solver is again the most expensive method by far. Its asymptotic convergence is similar to the other methods when ε_α is small enough.

4.1.3. Random walk on a Triangular Lattice. The last test problem appears in [34], and is taken from [32]. It considers a random walk on an $(m+1) \times (m+1)$ triangular grid, as shown in Figure (4.2) for $m = 6$. The points of the grid are labeled $(j, i), (i = 0, ..., m; j = 0, ..., m - i)$. From the point (j, i), a transition may take place to one of the four adjacent points $(j \pm 1, i \pm 1)$. The probability of jumping to either of the nodes $(j - 1, i)$ or $(j, i - 1)$ is $(\frac{j+i}{m})$, with the probability split equally between the two nodes when both are on the grid. The probability of jumping to either of the nodes $(j + 1, i)$ or $(j, i + 1)$ is $(1 - \frac{j+i}{m})$, with the probability again split equally when both nodes are on the grid. The spectrum of this matrix is real except for a few complex eigenvalues. This is one of the most difficult cases we tested with our smoothed aggregation solver.

size	lev	C_{op}	$u_{set}(t_{set})$	$algorithm$	V_{EIS}, V_{SOL}	γ	$u_{sol}(t_{sol})$
65536	7	1.95	23 (0.37s)	EIS	34 , 0	0.60	173(2.72s)
				AFTER(10^{-2})	1 , 38	0.63	46 (0.71s)
				AFTER(10^{-3})	4 , 27	0.58	50 (0.75s)
				AFTER(10^{-4})	7 , 23	0.59	63 (0.96s)
				AFTER(10^{-5})	11 , 21	0.58	81 (1.24s)
				OnTheFly(10^{-4})	2 , 27	0.58	41 (0.63s)
262144	8	1.96	22 (1.21s)	EIS	39 , 0	0.65	215(12.3s)
				AFTER(10^{-2})	1 , 63	0.82	72 (3.90s)
				AFTER(10^{-3})	4 , 53	0.76	80 (4.23s)
				AFTER(10^{-4})	7 , 30	0.65	70 (4.01s)
				AFTER(10^{-5})	12 , 25	0.64	91 (5.05s)
				OnTheFly(10^{-5})	2 , 36	0.64	52 (2.82s)

TABLE 4.3
Result Summary: Random walk on a Triangular Lattice

Table 4.3 shows the results for this problem. For the bigger problem, using the AFTER algorithm requires several setup cycles in order to reach the optimal asymptotic converge. Again, we see the tradeoff of the AFTER algorithm, as more setup cycles lead to a better convergence but require expensive computations. In the smaller problem, one additional setup is optimal for AFTER. As expected, using the EIS method by itself has the highest cost. The on-the-fly approach again achieves the best timings.

4.1.4. Discussion. We observe a high variability in the behavior of the SA solver with different strategies in different problems. In cases where the initial setup suffices for a reasonable convergence (even if not optimal), additional setup may not be cost-effective. This depends on the gain in performance that the additional setup provides compared to its cost. However, if additional setup cycles are needed, it is likely that the non-overlapping on-the-fly method is a better choice than AFTER.

4.2. Parallel environment simulation. In this section, we simulate a parallel environment using our serial implementation. We thus estimate the expected results of a comparison between the non-overlapping on-the-fly algorithm and the overlapping algorithm in a parallel environment when the computing resources are doubled. We again measure the expected performance in work-units, each equivalent to the cost of a single parallelized V_{SOL} cycle. As mentioned before, the more extensive parallelization results in a higher cost ratio between setup cycles and solution cycles. Therefore, the relative efficiency of parallel adaptive AMG algorithms deteriorates as the repeated setup cycles become more expensive (in work-units). In this section, we assume that we are already in a regime where adding more computing resources does not improve the cost of the multigrid cycles for a problem of fixed size.

We denote the run times of EIS and solution cycles in a serial environment by $time(V_{EIS}^{serial})$ and $time(V_{SOL}^{serial})$. These timings are measured in our serial code. $time(V_{EIS}^{parallel})$ and $time(V_{SOL}^{parallel})$ denote the corresponding timings in a parallel

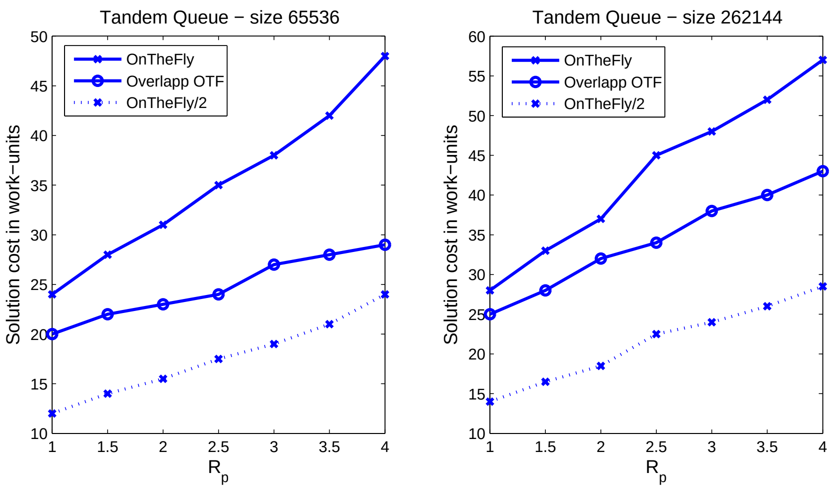

FIG. 4.3. *Tandem Queue: Parallel environment results*

environment. Next, we define

$$R_p = \frac{time(V_{EIS}^{parallel})/time(V_{SOL}^{parallel})}{time(V_{EIS}^{serial})/time(V_{SOL}^{serial})}, \tag{4.2}$$

which is the increase in the cost ratio between a setup and solution cycle as a result of the parallelization. The ratio R_p is expected to grow as we use more computing resources for a fixed problem size, because each computing node gets less grid points and hence communication is relatively more expensive.

We compare the non-overlapping and overlapping on-the-fly algorithms (Algorithms 4 and 5 respectively), considering again only the solution time beyond the first setup cycle, as this cycle is the same in both algorithms. The simulation of the non-overlapping on-the-fly algorithm is performed simply by multiplying the cost of V_{EIS} by R_p while keeping the cost of V_{SOL} fixed. For Algorithm 5 we need to simulate the procedure *overlapping-setup* in Section 2.3. We do this by multiplying the time it takes to perform an EIS cycle by R_p. This time (measured in work units) gives us the number of solution cycles carried out in parallel with the EIS cycle. We further assume that the communication required at each crossover costs the same as one solution cycle, so we add one work-unit to the cost of *overlapping-setup*. In the non-overlapping on-the-fly algorithm we use $\gamma = 0.75$ as before. In overlapping OTF we use a lower value of $\gamma = 0.5$, allowing more setup updates. The reason for this is that setup cycles constitute a smaller burden in this case because solution cycles are performed simultaneously.

In the following graphs we compare the estimated costs of the overlapping and non-overlapping OTF algorithms as a function of R_p in a parallel environment. Since we double the resources in overlapping OTF, the performance gain factor that can be achieved is bounded by 2. This is true also if we use double computing resources for the non-overlapping OTF. Therefore, as a reference, we also plot half the time of non-overlapping OTF (denoted by 'OTF/2').

4.2.1. Tandem Queue Markov Chain. Figure 4.3 shows the amount of work-units required for the solution as a function of R_p defined in Equation (4.2). The

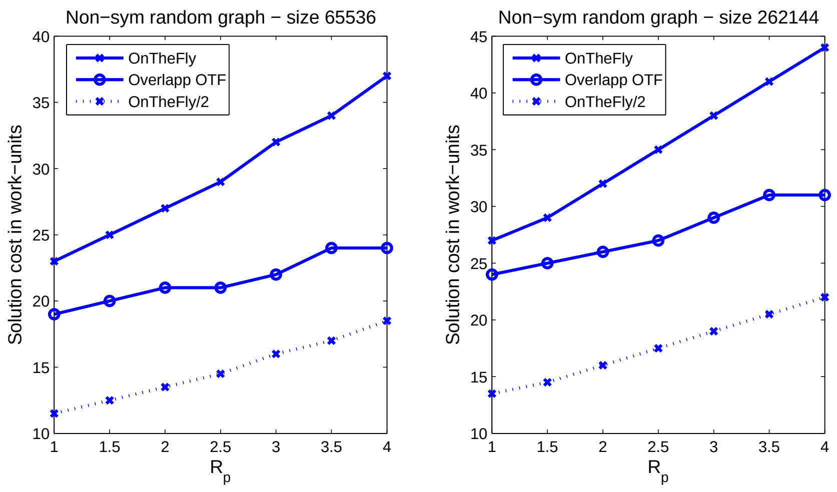

FIG. 4.4. *A random walk on a nonsymmetric random planar graph: Parallel environment results*

results indicate that the overlapping on-the-fly method has better performance as R_p increases. Indeed, the asymptotic slope of the overlapping OTF curve seems similar to that of the ideal OTF/2.

4.2.2. Random walk on a nonsymmetric random planar graph. Figure 4.4 shows the relevant results in work-units. Here, as in the results for $AFTER(10^{-2})$ shown in Table 4.2, one can see that the initial setup cycle suffices for achieving an adequate convergence factor. The overlapping on-the-fly algorithm takes full advantage of that, because the overlapping solution cycles are quite effective even though the operators are not very accurate yet. In both problem sizes we see that the solution cost increases very little with R_p.

Technically, the best option for this problem is to simply apply the initial setup and then continue with solution cycles with no additional EIS cycles. This approach is not influenced by the increasing R_p. However, as in the serial environment case, we cannot know this in advance so it is not a useful default choice for a general algorithm.

4.2.3. Random walk on a Triangular Lattice. The results are shown in Figure 4.5. Again, the overlapping on-the-fly algorithm achieves better performance than the non-overlapping version, although the performance gain is less significant than before. The asymptotic slope of the overlapping OTF is again similar to that of OTF/2.

4.2.4. Discussion. In a parallel computing environment the overlapping on-the-fly method has an advantage over its non-overlapping version, at the expense of the additional hardware that is required. We note again that if doubling the resources for an internally parallelized non-overlapping OTF significantly improves its performance, then that option would probably be a better choice than overlapping OTF. It would create curves that lie somewhere in between the OTF and OTF/2 curves, and that is also where the overlapping OTF curve lies. Once the resources are such that further internal parallelization is ineffective, the overlapping OTF approach can be employed to advantage.

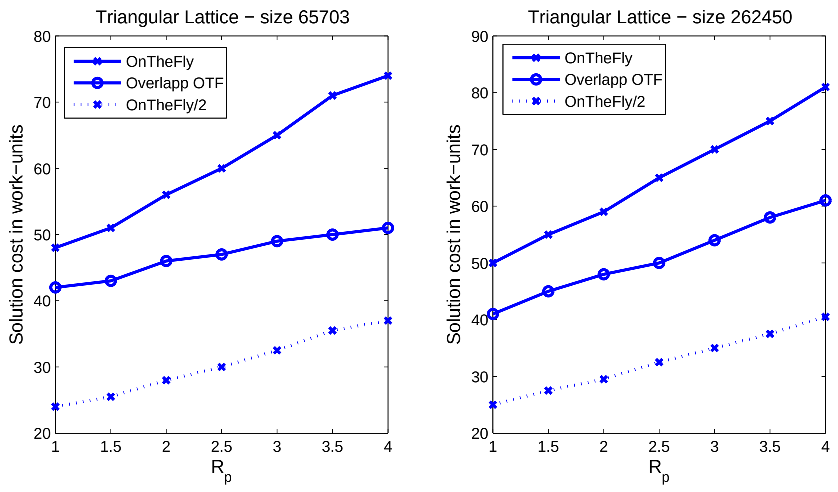

FIG. 4.5. *A random walk on a on a triangular lattice: parallel environment results*

5. Conclusions and Future Work.

This paper introduces an on-the-fly adaptive multigrid approach of interleaving adaptive EIS and classical multigrid cycles for Markov chain problems. The new idea is demonstrated using adaptive Petrov-Galerkin smoothed aggregation, where only the prolongation is smoothed. For the problems considered, using classical solution cycles instead of EIS cycles significantly reduced the costs of the solution process. The on-the-fly approach was generally found to be superior to applying solution cycles after a fixed number of EIS setup cycles.

A strategy more suitable to a parallel environment was also suggested, applying overlapping setup and solution cycles. This strategy seems to have potential in parallel computation, in regimes where the effectiveness of traditional parallelization has been exhausted, and yet additional resources are available.

Both on-the-fly approaches, with some modifications, may potentially be useful for enhancing the setup process of adaptive AMG methods for solving inhomogeneous linear systems, since their setup usually targets a homogenous system.

REFERENCES

[1] M. Bolten, A. Brandt, J. Brannick, A. Frommer, K. Kahl, and I. Livshits, *A bootstrap algebraic multilevel method for markov chains*, Submitted to SIAM J. Sci. Comput., (2010).
[2] A. Borzi and G. Borzi, *Algebraic multigrid methods for solving generalized eigenvalue problems*, Int. J. Numer. Meth. Eng., 65 (2006), pp. 1186–1196.
[3] A. Brandt, S. McCormick, and J. Ruge, *Multigrid methods for differential eigenproblems*, SIAM. J. Stat. Sci. Comput., 4 (1983), pp. 655–684.
[4] 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.
[5] A. Brandt and D. Ron, *Multigrid solvers and multilevel optimization strategies*, in Multilevel Optimization and VLSICAD, Kluwer (Boston), 2003, pp. 1–69.
[6] 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.
[7] ———, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.
[8] M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, and G. Sanders, *Towards adaptive smooth aggregation (α SA) for nonsymmetric problems*, SIAM J. Sci. Comput., 32 (2010), pp. 14–39.

[9] 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.

[10] J. E. Dendy, *Black box multigrid*, J. Comput. Phys., 48 (1982), pp. 366–386.

[11] M. Emans, *Efficient parallel amg methods for approximate solutions of linear systems in cfd applications*, SIAM J. Sci. Comput., 32 (2010), p. 22352254.

[12] H. Guillard, A. Janka, and P. Vaněk, *Analysis of an algebraic petrov–galerkin smoothed aggregation multigrid method*, Applied Numerical Mathematics, 58 (2008), pp. 1861–1874.

[13] H. Guillard and P. Vaněk, *An aggregation multigrid solver for convection-diffusion problems onunstructured meshes.*, Tech. Report UCD-CCM-130, University of Colorado at Denver, CO, USA, 1998.

[14] 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.

[15] 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.

[16] 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.

[17] 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).

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

[19] J. J. Heys, T. A. Manteuffel, S. F. McCormick, and L. N. Olson, *Algebraic multigrid for higher-order finite elements*, J. Comput. Phys., 204 (2005), pp. 520–532.

[20] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, 1985.

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

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

[23] A. C. Muresan and Y. Notay, *Analysis of aggregation-based multigrid*, SIAM J. Sci. Comput., 30 (2008), pp. 1082–1103.

[24] Y. Notay, *Aggregation-based algebraic multilevel preconditioning*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 998–1018.

[25] ———, *An aggregation-based algebraic multigrid method*, Electronic Transactions on Numerical Analysis, to appear, (2010).

[26] Y. Notay and P. S. Vassilevski, *Recursive krylov-based multigrid cycles*, Numerical Linear Algebra With Applications, 15 (2008), p. 473487.

[27] J. Ruge, *Algebraic multigrid (AMG) for geodetic survey problems*, in Proceedings of the International Multigrid Conference, Copper Mountain, CO, 1983.

[28] 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.

[29] M. Sala and R. S. Tuminaro, *A new petrov-galerkin smoothed aggregation preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 31 (2008), pp. 143–166.

[30] H. D. Sterck, K. Miller, G. Sanders, E. Treister, and I. Yavneh, *Fast multilevel methods for markov chains*, In preperation, (2010).

[31] H. D. Sterck, U. M. Yang, and J. J. Heys, *Reducing complexity in parallel algebraic multigrid preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1019–1039.

[32] G. Stewart, *Matrix generator MVMRWK*. http://math.nist.gov/MatrixMarket/data/NEP/ mvmrwk/ mvmrwk.html.

[33] K. Stüben, *Algebraic multigrid (amg): experiences and comparisons*, Appl. Math. Comput., 13 (1983), pp. 419–451.

[34] E. Treister and I. Yavneh, *Square and stretch multigrid for stochastic matrix eigenproblems*, Numerical Linear Algebra with Application, 17 (2010), pp. 229–251.

[35] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Num. Math., 88 (2001), pp. 559–579.

[36] P. Vaněk, M. Brezina, and R. Tezaur, *Two-grid method for linear elasticity on unstructured meshes*, SIAM J. Sci. Comput., 21 (1999), pp. 900–923.

[37] P. Vaněk, A. Janka, and H. Guillard, *Convergence of algebraic multigrid based on smoothed aggregation ii: Extension to a petrov-galerkin method*, tech. report, 1999.

[38] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[39] E. Virnik, *An algebraic multigrid preconditioner for a class of singular M-matrices*, SIAM J. Sci. Comput., 29 (2007), pp. 1982–1991.

[40] U. M. Yang, *Parallel algebraic multigrid methods–high performance preconditioners*, in Numerical Solutions of Partial Differential Equations on Parallel Computers, Lect. Notes Comput. Sci. Eng. 51, A. M. Bruaset and A. Tweiter eds., Springer-Verlag, Berlin, 2006, pp. 209–236.