Adv Comput Math (2011) 35:375–403
DOI 10.1007/s10444-010-9168-x

Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains

H. De Sterck · K. Miller · T. Manteuffel · G. Sanders

Received: 8 October 2009 / Accepted: 15 April 2010 /
Published online: 17 June 2010

Abstract In many application areas, including information retrieval and networking systems, finding the steady-state distribution vector of an irreducible Markov chain is of interest and it is often difficult to compute efficiently. The steady-state vector is the solution to a nonsymmetric eigenproblem with known eigenvalue, $B\mathbf{x} = \mathbf{x}$, subject to probability constraints $\|\mathbf{x}\|_1 = 1$ and $\mathbf{x} > \mathbf{0}$, where B is column stochastic, that is, $B \geq O$ and $\mathbf{1}^t B = \mathbf{1}^t$. Recently, scalable methods involving Smoothed Aggregation (SA) and Algebraic Multigrid (AMG) were proposed to solve such eigenvalue problems. These methods use multiplicative iterate updates versus the additive error corrections that are typically used in nonsingular linear solvers. This paper discusses an outer iteration that accelerates convergence of multiplicative update methods, similar in principle to a preconditioned flexible Krylov wrapper applied to an additive iteration for a nonsingular linear problem. The acceleration is performed by selecting a linear combination of old iterates to minimize a functional within the space of probability vectors. Two different implementations of this idea are considered and the effectiveness of these approaches is demonstrated with representative examples.

Communicated by Rafael Bru and Juan Manuel Peña.

H. De Sterck · K. Miller
Department of Applied Mathematics, University of Waterloo,
Waterloo, ON, Canada N2L 3G1

T. Manteuffel · G. Sanders (✉)
Department of Applied Mathematics, University of Colorado, 526 UCB,
Boulder, CO 80309-0526, USA
e-mail: sandersg@colorado.edu

Keywords Algebraic multigrid · Smoothed aggregation · Markov chain · Acceleration

Mathematics Subject Classifications (2010) 65C40 · 60J22 · 65F10 · 65F15

1 Introduction

This work develops a technique to accelerate multiplicative multilevel methods that calculate the stationary probability vector of large, sparse, irreducible, slowly-mixing Markov chains. Large sparse Markov chains are of interest in a wide range of applications, including information retrieval and web ranking, performance modelling of computer and communication systems, dependability and security analysis, and analysis of biological systems [18].

A Markov chain with n states is represented by an $n \times n$ non-negative matrix, B, that is column-stochastic, $\mathbf{1}^t B = \mathbf{1}^t$. The stationary vector that we seek, $\mathbf{x}$, satisfies the following eigenproblem with known eigenvalue:

$$B\mathbf{x} = \mathbf{x}, \qquad \|\mathbf{x}\|_1 = 1, \qquad \mathbf{x} \geq 0, \tag{1.1}$$

where the normalization constraint and the non-negativity constraint make $\mathbf{x}$ a *probability vector*. If any state in the chain is connected to any other state through a series of directed arcs, then the matrix B is called *irreducible*. We assume this property, which guarantees that there is a unique solution to (1.1), which is strictly positive ($\mathbf{x} > \mathbf{0}$), by the Perron–Frobenius theorem (see [1, 15] for details).

The power method converges to $\mathbf{x}$ when B is *aperiodic*, meaning the length of all directed cycles on the graph of B have greatest common denominator equal to one. However, the rate of convergence of the power method, and similar one-level iterative methods like Jacobi and Gauss-Seidel, is dependent on the *magnitude of the subdominant eigenvalue(s)*, which we denote

$$|\lambda_2| := \max |\lambda| \quad \text{for} \quad \lambda \in \Sigma(B) \setminus \{1\}.$$

When $|\lambda_2| \approx 1$, B is called *slowly-mixing*, and the convergence rates of classical iterative techniques are unacceptably close to 1 as well. For many Markov chains of interest, $|\lambda_2| \to 1$ as the problem size increases and these classical iterative techniques are not algorithmically scalable for such problems. An *algorithmically scalable* algorithm would achieve approximate solutions up to a given tolerance with an amount of work proportionate to the amount of information in the problem matrix B (which for the problems we consider is proportional to the number of unknowns). For many of these problems, applying Krylov acceleration (such as preconditioned GMRES [12]) to classical iterative methods does not mend the scalability. This is because the inherent local influence of these techniques requires a high number of iterations to properly realize the desired global distribution from a poorly distributed initial guess.

Eigenproblem (1.1) is equivalent to the following singular linear problem:

$$A\mathbf{x} = \mathbf{0}, \qquad \|\mathbf{x}\|_1 = 1, \qquad \mathbf{x} \geq 0, \tag{1.2}$$

where $A := I - B$. This formulation has some specific advantages. First, the vector we seek, $\mathbf{x}$, is the right eigenvector of A corresponding to eigenvalue 0, and also is a *right singular vector* of A corresponding to singular value 0. Vector $\mathbf{x}$, however, is not necessarily a right singular vector corresponding to B. Advantages of working with the singular value decomposition of A are given in Section 3, where we discuss how to form minimization principles for accelerating multilevel methods.

Another important advantage of working with problem (1.2) is that the *M-matrix structure* of A (which implies $a_{ii} > 0$ and $a_{ij} \leq 0$ for $i \neq j$) is amenable to additive Algebraic Multigrid (AMG) solvers designed for nonsingular linear problems [3, 4]. See [7, 20] for an introduction to AMG. Here, a fixed multigrid hierarchy is first built and then applied to find a nontrivial solution to the linear problem $A\mathbf{x} = \mathbf{0}$. In [22], AMG-preconditioned Krylov acceleration was employed. These standalone and accelerated versions of AMG rely on the fixed multigrid hierarchy being able to adequately approximate any vector $\mathbf{e}$ that is *near-kernel*, $\|A\mathbf{e}\| << \|\mathbf{e}\|$. If this assumption is not met, then algorithmic scalability of the method is not achieved.

Many techniques designed to adaptively create multigrid hierarchies for solving nonsingular linear systems are based on a standard AMG approach [6] and a smoothed aggregation (SA) approach [5]. The setup phases of these methods essentially adjust the multilevel hierarchies so that near-kernel vectors of A are adequately approximated by coarse grids.

Several closely related adaptive multiplicative techniques have been designed to solve (1.2) directly. In an unsmoothed aggregation form (also called aggregation/disaggregation) [9, 13, 15, 18], algorithmic scalability is not achieved for many problems due to poor approximation of the kernel of A by unsmoothed intergrid operators. More recent multiplicative methods employ hierarchies with richer representation of the kernel and demonstrate scalability. These include a method using SA hierarchies [16], one using AMG hierarchies [14], and a newer method that uses an unsmoothed aggregation hierarchy on a variant of the squared problem, $B^2\mathbf{x} = \mathbf{x}$ [19].

These multiplicative schemes use multilevel hierarchies that adapt with every cycle. Therefore, a standard Krylov acceleration technique cannot be applied, because the spaces involved are not related by a fixed preconditioner applied to residual vectors. However, *flexible* acceleration is possible for methods with changing hierarchies or nonstationary preconditioners (FGMRES [11] is a common example of this). In this paper, we do not use flexible GMRES or flexible CG, but we present two acceleration techniques that are customized to solve problem (1.2). The first technique employs an unconstrained minimization problem and the second technique employs a constrained minimization problem. We briefly show that both minimization problems have a unique probability solution that is the stationary probability vector. This work demonstrates the application of the acceleration techniques

to versions of the classical unsmoothed aggregation algorithm [9, 15], the SA algorithm given in [16], and the AMG algorithm in [14].

Similar accelerators have been designed for other nonlinear iterations. In [23], an accelerator is developed for the multilevel Fast Approximation Scheme (FAS [2], see [7] for an introduction), which is used to solve nonlinear PDEs. A key difference is that the accelerator for Markov problems must produce probability vectors, a feature not required for general nonlinear problems. Another difference is that the acceleration of FAS for nonlinear problems requires linearization of target functionals, but our multiplicative approach does not rely on linearization. The FAS accelerator does share many characteristics of the accelerators we develop here, including use of similar minimization functionals.

This paper is organized as follows. Section 2 provides some background on the two methods we consider accelerating [14, 16] and some minor enhancements to these methods. Section 3 describes a framework for accelerating nonlinear iterative methods that solve (1.2). Section 4 describes a specific approach to acceleration that uses an unconstrained minimization problem. Section 5 describes another approach to acceleration that uses a constrained minimization problem. Section 6 contains numerical results that highlight the performance of the acceleration and Section 7 provides concluding remarks.

2 Background: adaptive multiplicative multilevel methods

In this section, we first present a general framework for the class of multilevel methods for which the acceleration techniques developed in this paper apply. In the subsections that follow, we mention the specific methods in this class that are tested in this work.

Consider relaxation techniques of the form $\mathbf{x} \leftarrow (I - M^{-1}A)\mathbf{x}$, where M^{-1} is an inexpensive and locally computable preconditioner. Many classical iterative methods fit into this category; the power, Jacobi, and Gauss-Seidel methods are all examples. In this paper, we use weighted-Jacobi iteration for relaxation,

$$\mathbf{x} \leftarrow (I - \alpha D^{-1}A)\mathbf{x}, \tag{2.1}$$

with $\alpha = 0.7$. Such relaxation techniques are cheap per iteration but quickly stall, having little effect on the *right near kernel* of $M^{-1}A$, defined to be any vector $\mathbf{e}$ such that

$$\|M^{-1}A\mathbf{e}\|_2 << \|\mathbf{e}\|_2.$$

These techniques have no effect on the *actual kernel* of $M^{-1}A$, as is desired to solve problem (1.2). However, for many example problems, there is a rich class of vectors $\mathbf{y} \neq \mathbf{x}$ that are also near-kernel, or vectors that the relaxation alone does not efficiently remove. Such vectors are referred to as *algebraically smooth* within the AMG literature. Essentially, these iterative schemes quickly produce vectors with the local character of the solution, $\mathbf{x}$,

but take many iterations before the iterates have the global character. In this situation, multilevel techniques are commonly employed to complement the relaxation method by efficiently producing iterates that have both local and global qualities of the solution vector.

We consider multilevel techniques designed in the algebraic multigrid (AMG) framework, where a hierarchy of coarse grids is developed based only on the size and structure of the entries of the matrix A, instead of relying on the geometry of the original problem. This is appropriate for problems that arise from Markov chains, as there is typically no underlying geometry, or it is sufficiently complicated to warrant a more automatic coarsening routine.

A multilevel hierarchy is a data structure containing *problem matrices*, *intergrid transfer operators*, *relaxation techniques*, usually stored in the form of preconditioners, and a *coarsest-level solver*. The *level* of the hierarchy is an integer $l = 1, 2..., L$, where the *coarsest level* is $l = L$ and the *finest level* is $l = 1$. The problem matrices, A_l, are singular M-matrices of size $n_l \times n_l$ and the size of these matrices decreases rapidly per level, $n_l > n_{l+1}$. Note that the finest-level problem matrix, A_1, is the matrix from the original problem (1.2) and the coarsest-level problem matrix, A_L, is a small $n_L \times n_L$ matrix. There are two types of intergrid transfer operators: *restriction*, R_l^{l+1}, which maps vectors from the fine level $\mathbb{R}^{n_l}$ to the coarse level $\mathbb{R}^{n_{l+1}}$, and *interpolation*, P_{l+1}^l, which maps vectors from the coarse level to the fine level. The relaxation technique for a certain level is typically represented by a simple preconditioner based on the problem matrix of that level. Additionally, a solver for the coarsest-level problem, $A_L \mathbf{x}_L = \mathbf{0}_L$ is specified. In summary, the full multilevel hierarchy is the following set,

$$\left\{ A_l,\ R_l^{l+1},\ P_{l+1}^l,\ M_l^{-1} \right\}_{l=1}^{L-1} \quad \cup \quad \{\text{a solver for } A_L \mathbf{x}_L = \mathbf{0}_L\}. \tag{2.2}$$

See the left part of Fig. 1 for a visual representation of when various levels are employed in the typical multigrid V-cycle.

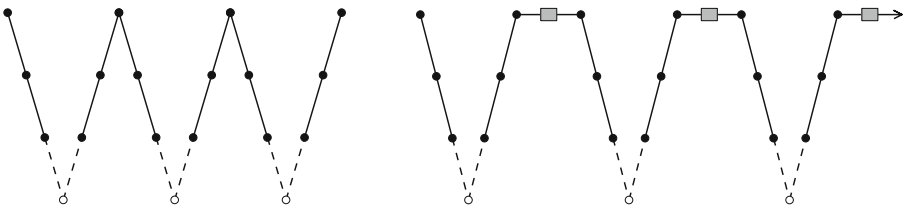

Fig. 1 Diagram of standalone V-cycles and Accelerated V-cycles. On the *left* is the standalone algorithm, which proceeds from *left* to *right*. Fine-grid operations are represented at the *top of the diagram*, coarse-grid operations are on the *bottom*, and intermediate-grid operations are *in between*. The *black dots* represent relaxation operations on their respective grids and the *open dots* represent coarse-level solves. On the *right is a diagram* of an accelerated V-Cycle, where an acceleration step, represented by a *gray box*, is added at the end of each cycle

For a method in the class of *adaptive multiplicative* multilevel methods, this hierarchy is not static. (It typically is static after the initial setup of AMG applied to a nonsingular linear system of equations). Instead, as the algorithm progresses, the members of the multilevel hierarchy are adapted to achieve a more and more accurate representation of the near kernel of A_1. Each cycle is also *multiplicative*, meaning the iterates that the method produces come directly from the range of the changing interpolation operators (as opposed to an *additive* error correction coming from the range of interpolation).

For the rest of this section we use two-level notation to describe the interaction of only two grids at a time. Generally, the problem matrix on the current level, A_l, is represented by A and the matrix on the next coarser level, A_{l+1}, is represented by A_c. The notation is the same for several other types of objects: those objects that pertain to the current grid have no subscript or the subscript f and those that pertain to the next coarser level have the subscript c. The symbols representing intergrid transfer operators involved in levels l and $l+1$, restriction, R, and interpolation, P, have neither subscripts nor superscripts.

Intergrid transfer operators are designed to create a coarse-grid problem that accurately represents left and right near kernel components of A. The actual left-kernel vector of A is known to be the constant vector, $\mathbf{1}$, so R can be chosen such that representation of actual left kernel within the range of R^t is fully accurate. Ideally, the range of P contains the actual right kernel as well. However, the right-kernel vector, $\mathbf{x}$, is not known (it is the target of the method) and a fully accurate representation of $\mathbf{x}$ is not guaranteed. Heuristically, the intergrid transfer operators are formed to have properties

$$\mathbf{1} \in \mathcal{R}(R^t) \qquad \text{and} \qquad \mathbf{x} \,\tilde{\in}\, \mathcal{R}(P), \tag{2.3}$$

where $\tilde{\in}$ is to be interpreted loosely as *"is approximately in the range of"*. The additional requirement of sparsity is necessary as well. The strategy for forming P is based on the idea that a low number (one or two) of relaxations produces a vector $\mathbf{x}_k$ that is locally similar to the right kernel of A. Then, an interpolation operator is formed that exactly represents the relaxed vector at the k-th iteration, $\mathbf{x}_k \in \mathcal{R}(P)$. In practice, P is constructed such that $\mathbf{x}_k = P\mathbf{1}_c$, where $\mathbf{1}_c$ is the constant vector on the coarse grid. The global character of the approximate right-kernel vector may be adjusted on a coarser (and therefore cheaper) grid, $\mathbf{x} \approx P\mathbf{x}_c$, where $\mathbf{x}_c$ is some coarse-grid representation of the actual right kernel of the coarse-level problem (which is a representation of the multiplicative error on the coarse level). Two specific approaches for choosing the structure of R and P are presented in Sections 2.1 and 2.2.

When R and P have been formed, we have the following coarse-grid problem

$$A_c\mathbf{x}_c = \mathbf{0}_c, \qquad \|\mathbf{x}_c\|_1 = 1, \qquad \mathbf{x}_c > 0, \tag{2.4}$$

where the coarse-grid problem matrix $A_c = RAP$, and $\mathbf{0}_c$ is the vector of all zeros on the coarse grid. Under this construction, the following properties are preserved on all grids,

$$\begin{aligned} \mathbf{1}_c^t A_c &= \mathbf{0}_c \ \forall \mathbf{x}_k, \\ A_c \mathbf{1}_c &= \mathbf{0}_c \text{ for } \mathbf{x}_k = \mathbf{x}. \end{aligned} \tag{2.5}$$

Additionally, we require that the M-matrix structure and irreducibility are preserved on all grids. Section 2.3 introduces a *lumping* strategy that is used to ensure these properties.

In the following two subsections, we briefly discuss the versions of Algorithm 1 that are accelerated in this paper. The algorithm presented in [19] fits into a closely related framework where no lumping step is necessary. Although no tests were done here, similar acceleration techniques should be applicable.

Algorithm 1: Algebraic Multilevel Method for Markov chains (γ-cycle)
$\mathbf{x} \leftarrow$**AMMM**$(A, \mathbf{x}, \nu_1, \nu_2, \gamma)$

1. Pre-relax, $\mathbf{x} \leftarrow (I - M^{-1}A)^{\nu_1}\mathbf{x}$.
2. Choose R and P to satisfy (2.3).
3. Form $A_c = RAP$. If necessary, form $\hat{A}_c$ by lumping (see Section 2.3).
4. If n_c is small enough then solve $\hat{A}_c\mathbf{x}_c = 0$ directly, otherwise solve it approximately by initializing $\mathbf{x}_c \leftarrow \mathbf{1}_c$ and then doing the following γ times:

$$\mathbf{x}_c \leftarrow \mathbf{AMMM}(\hat{A}_c, \mathbf{x}_c, \nu_1, \nu_2, \gamma)$$

5. Interpolate, $\mathbf{x} \leftarrow P\mathbf{x}_c$.
6. Post-relax, $\mathbf{x} \leftarrow (I - M^{-1}A)^{\nu_2}\mathbf{x}$
7. Normalize, $\mathbf{x} \leftarrow \mathbf{x}/\|\mathbf{x}\|_1$

The next two subsections describe particular choices of R, P, and A_c used by the algebraic multilevel methods that are accelerated in this work.

2.1 Aggregation multigrid methods for Markov chains

This section describes the first class of methods we consider , where the structure of our intergrid transfer operators is given by an *aggregation*, a grouping of degrees of freedom based on a strength-of-connection measure [15, 16]. For the aggregation methods, we use a strength-of-connection measure that is dependent on the current iterate. Let diag($\mathbf{x}_k$) be the diagonal matrix with the entries of $\mathbf{x}_k$ on its diagonal. Node i is considered to be *strongly connected to* node j in the graph of A diag($\mathbf{x}_k$) if

$$-a_{ij}x_j \geq \theta \max_{p \neq i}\{-a_{ip}x_p\} \quad \text{or} \quad -a_{ji}x_i \geq \theta \max_{p \neq j}\{-a_{jp}x_p\}, \tag{2.6}$$

where x_p denotes the p-th entry of the current iterate $\mathbf{x}_k$. Note that this is a symmetrized strength-of-connection measure that is weighted by the current

iterate. For a given *strength-of-connection parameter*, θ, define $\mathcal{N}_i$ to be the *strong neighborhood* of i, which contains i and any $j \neq i$ such that at least one of the relationships in (2.6) is satisfied. We believe that some improvements could be made to our definition of strength-of-connection, especially for matrices with highly nonsymmetric entries and sparsity patterns. However, we use this symmetrized definition in our current implementation.

An aggregation is a disjoint partition of unity that is represented by an $n \times n_c$ binary matrix, Q. Each column of this matrix corresponds to an aggregate and each row corresponds to a fine-level degree of freedom. If entry $q_{ij} = 1$, then fine-grid degree of freedom i belongs to aggregate j.

In this paper, matrix Q is computed using strength-of-connection measure (2.6) and a *neighborhood-based* aggregation technique given in [21]. Note that this aggregation is related to common versions of aggregation in the AMG literature and differs from the aggregation techniques used in [15, 16], called *distance-1* and *distance-2 aggregation*. See Fig. 2 for a visual example of neighborhood-based aggregation versus distance-2 aggregation. The distance-d aggregation techniques do not ensure that a proper strongly-connected neighborhood of points is contained in each aggregate, leading to aggregates that vary greatly in size (aggregate sizes vary from 1 to 12 points in the example in Fig. 2). This discrepancy in aggregate size leads to poorer coarse-grid approximation properties and larger coarse-grid stencil sizes, particularly in the smoothed aggregation case. Alternatively, the neighborhood-based aggregation ensures that each aggregate contains at least a proper neighborhood and the size of each aggregate is much closer to the stencil sizes involved in the graph of A (aggregate sizes vary from 3 to 7 points in the example in Fig. 2). This enforces a more regular coarsening throughout the domain, which provides better sparsity on coarse grids without losing approximation accuracy of near-kernel vectors.

Algorithm 2: Neighborhood-Based Aggregation, $\{Q_j\}_{j=1}^{J} \leftarrow$**Agg**$(A, \mathbf{x}, \theta)$

Set $\mathcal{R} \leftarrow \{1, ..., n\}$ and $J \leftarrow 0$.
`/* 1st pass: assign entire neighborhoods to aggregates */`
for $i \in \{1, ..., n\}$ **do**
  Define $\mathcal{N}_i$ based on (2.6).
  if $\mathcal{N}_i \subset \mathcal{R}$ **then** `/* if entire n'hood is unassigned */`
    $J \leftarrow J + 1$.
    $\hat{Q}_J = \mathcal{N}_i$. `/* assign n'hood to a new aggregate */`
    $\mathcal{R} \leftarrow \mathcal{R} \setminus \mathcal{N}_i$.
  end
end
for $j = 1, ..., J$ **do** $Q_j \leftarrow \hat{Q}_j$.
`/* 2nd pass: put remaining pts in most connected aggregates */`
while $\mathcal{R} \neq \emptyset$ **do**
  Pick $i \in \mathcal{R}$ and set $j = \text{argmax}_{k=1,...,J}\, \text{card}(\mathcal{N}_i \cap \hat{Q}_k)$.
  Set $Q_j \leftarrow Q_j \cup \{i\}$ and $\mathcal{R} \leftarrow \mathcal{R} \setminus \{i\}$.
end

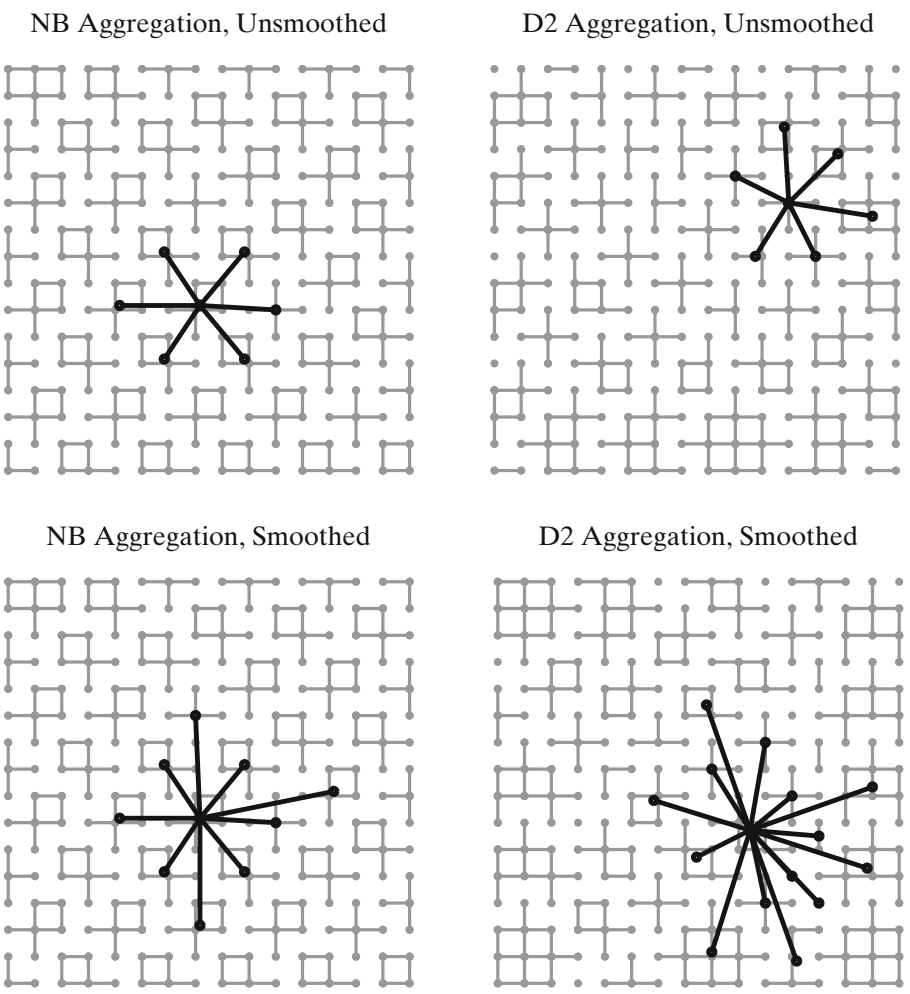

Fig. 2 Neighborhood-based aggregation versus distance-2 aggregation. In each diagram above, aggregations were performed on a standard 2D graph Laplacian of a 16×16 mesh with a five-point stencil at each interior node. The *left column* shows neighborhood-based aggregations and the *right column* shows distance 2 aggregations. The groups involved in each aggregation are displayed as sets of *gray dots connected by gray lines*. Additionally, a coarse-grid stencil of largest size is displayed with *black dots connected by thick black lines*. Stencils for unsmoothed aggregation are displayed in the *top row* and stencils for smoothed aggregation are displayed in the *bottom row*

For an unsmoothed aggregation method, the intergrid transfer operators are set to $R = Q^t$ and $P = \text{diag}(\tilde{\mathbf{x}})\,Q$, where $\tilde{\mathbf{x}}$ is the most recent approximate solution that has been relaxed.

2.1.1 Smoothing intergrid transfer operators

The representation of the left near kernel and right near kernel within the range of Q^t and $\text{diag}(\mathbf{x})\,Q$, respectively, is often greatly improved by applying

a smoothing operator to these intergrid transfer operators. Additionally, the representation of the algebraically oscillatory vectors is reduced by smoothing. In the context of an additive solver for a nonsingular problem, the latter is arguably the greater impact of smoothing, as it greatly increases the efficiency of relaxation on each coarse grid.

If the error propagation operator of the relaxation process is sparse, then some version of it is used for smoothing the intergrid transfer operators. Although this typically causes the multigrid hierarchy to have more computational complexity, smoothing the intergrid transfer operators often makes the aggregation method scalable. This has been observed both for nonsingular linear problems in [21] and steady-state Markov eigenproblems in [16]. For aggregation methods, the intergrid transfer operators are set to

$$R = Q^t(I - \alpha_R A D^{-1}) \quad \text{and} \quad P = (I - \alpha_P D^{-1} A)\text{diag}(\tilde{\mathbf{x}})\, Q, \tag{2.7}$$

where (α_R, α_P) are smoothing parameters. The following choices for (α_R, α_P) give the various intergrid transfer operators smoothing used in this work:

$$\begin{array}{ccc} \text{smoothed aggregation,} & \text{smooth } P \text{ only,} & \text{unsmoothed aggregation.} \\ (\alpha_R, \alpha_P) = (0.7, 0.7) & (\alpha_R, \alpha_P) = (0, 0.7) & (\alpha_R, \alpha_P) = (0, 0) \end{array} \tag{2.8}$$

2.2 Algebraic multigrid for Markov chains

In the second class of methods we consider accelerating, MCAMG [14], the intergrid transfer operators are based on the principles of classical additive AMG for nonsingular linear systems.

We first perform a variant of the AMG coarsening routine described in [7], which determines the structure of the intergrid transfer operators. Strength-of-connection is based on the scaled fine-level operator,

$$-a_{ij}x_j \geq \theta \max_{k \neq i}\{-a_{ik}x_k\}, \tag{2.9}$$

with strength-of-connection parameter $\theta \in (0, 1)$. Note that the symmetric strength-of-connection measure we use for aggregation methods, (2.6), differs from the nonsymmetric one we use here, (2.9). Define N_i to be the *directed strong neighborhood* of point i, which contains any $j \neq i$ such that the relationship in (2.9) is satisfied. Notice that N_i differs from the $\mathcal{N}_i$ defined in the previous section.

Using this strength-of-connection measure, the fine set of degrees of freedom, $H = \{1, 2, ..., n_f\}$ is partitioned into two sets using two-pass Ruge-Stuben coarse-fine splitting. Formally, this means $H = C \cup F$, where the set of n_c *coarse points* is C and the set of $(n_f - n_c)$ *fine points* is F. See [7, 14] for the exact coarse-fine splitting algorithm.

After the splitting is performed, we define the structure of intergrid transfer operators using a variant of the classical AMG interpolation formula. Instead of an aggregation matrix, for Q we use an *overlapping partition of unity*, with the properties $1 \geq Q \geq 0$ and $\mathbf{1}_f = Q\mathbf{1}_c$. For any $i \in F$, define C_i to be the set

of C-points strongly influencing point i, in the sense of (2.9). The structure of the entries in Q is

$$(Q\mathbf{e}_c)_i = \begin{cases} (\mathbf{e}_c)_i & \text{if } i \in C, \\ \sum_{j \in C_i} w_{ij} (\mathbf{e}_c)_j & \text{if } i \in F. \end{cases}, \tag{2.10}$$

where $\mathbf{e}_c$ is any coarse-level vector of size n_c and $Q\mathbf{e}_c$ is its image, a vector of size n_f. Coefficients w_{ij} are *interpolation weights* that are determined by the following formula:

$$w_{ij} = \frac{a_{ij}x_j + \sum_{m \in D_i^s} \left(\frac{a_{im}x_m a_{mj}x_j}{\sum_{k \in C_i} a_{mk}x_k} \right)}{\sum_{l \in C_i} a_{il}x_l + \sum_{r \in D_i^s} a_{ir}x_r}, \tag{2.11}$$

with decomposition $N_i = C_i \cup D_i^s \cup D_i^w$, where D_i^s is the set of points in N_i that strongly influence i, and D_i^w is the set of points in N_i that do not strongly influence i. Under this construction and the assumptions on a_{ij}, we are guaranteed nonnegative interpolation weights, w_{ij}. Again, see [14] regarding the motivation for (2.11). After the overlapping partition of unity, Q, is computed we specify restriction and interpolation to be

$$R = Q^t \qquad \text{and} \qquad P = \text{diag}(\tilde{\mathbf{x}})\, Q.$$

2.3 Lumping coarse operators

In order to guarantee that coarse-grid problem (2.4) has a unique and positive solution, we may have to perturb A_c by small amounts. We use a result from [1] (presented in this context in [16]) that uses singular M-matrix results from Perron-Frobenius theory. Matrix A is a *singular M-matrix* if and only if there is a non-negative matrix B such that $A = \rho(B)I - B$, where $\rho(B)$ is the spectral radius of B. The fine-grid matrix, A, fits this definition by construction. We use two results to ensure that the coarse-grid matrices fit this definition as well. First, if a coarse-grid matrix is irreducible, has non-positive off-diagonal elements, and a strictly positive left-kernel vector then it is an irreducible singular M-matrix. Second, if a coarse-grid matrix is an irreducible singular M-matrix, then there exists a strictly positive vector, unique up to a scaling, in the right kernel of this matrix.

We use the lumping technique from [14, 16] to ensure that correct sign structure and irreducibility are both maintained for the coarse-grid matrices. Matrix A is a singular M-matrix, so it has the splitting $A = D - C$, where $D \geq 0$ is the diagonal part of A and $C \geq 0$ is the off-diagonal part. Operators R and P are also non-negative, so

$$A_c = RAP = RDP - RCP =: S - G, \tag{2.12}$$

where $S \geq 0$ and $G \geq 0$. For the R and P selected by unsmoothed aggregation, S is diagonal and strictly positive on the diagonal, so it cannot produce positive

off-diagonal elements in A_c. Coarse-level operator A_c has a strictly positive left-kernel vector:

$$\mathbf{1}_c A_c = \mathbf{1}_c RAP = \mathbf{1}_f AP = \mathbf{0}.$$

The irreducibility of A_c is automatic for unsmoothed R and P (see [16]). This, combined with the correct sign structure of A_c and the positive left-kernel vector, implies that A_c is an irreducible singular M-matrix, and thus has a unique and strictly positive right-kernel vector as well, as summarized from [1] in the beginning of this section.

For the R and P selected by SA or AMG, matrix S is generally not diagonal, so there is no guarantee that $s_{ij} - g_{ij} \leq 0$ whenever $i \neq j$. Also, zeros can be produced in A_c where G is nonzero (thus possibly making A_c reducible, see [16]). To ensure our coarse-grid operator is irreducible and has the appropriate sign structure, small perturbations are added to S for any *offending pair* $\{i, j\}$ where $g_{ij} \neq 0$ and $s_{ij} - g_{ij} \geq 0$, or $g_{ji} \neq 0$ and $s_{ji} - g_{ji} \geq 0$. Initially, set $\hat{S} \leftarrow S$. Then, a first offending pair is found, and value $\beta_{\{i,j\}} \geq 0$ is chosen to satisfy

$$\begin{aligned} \hat{s}_{ij} - g_{ij} - \beta_{\{i,j\}} &\leq -\eta g_{ij}, \qquad \text{and} \\ \hat{s}_{ji} - g_{ji} - \beta_{\{i,j\}} &\leq -\eta g_{ji}, \end{aligned} \tag{2.13}$$

with a small *lumping parameter*, $\eta > 0$. The perturbation is given as

$$S_{\{i,j\}} = \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} & & i & & j \end{array} \\ \begin{bmatrix} \ddots & \vdots & & \vdots & \\ \cdots & \beta_{\{i,j\}} & \cdots & -\beta_{\{i,j\}} & \cdots \\ & \vdots & & \vdots & \\ \cdots & -\beta_{\{i,j\}} & \cdots & \beta_{\{i,j\}} & \cdots \\ & \vdots & & \vdots & \end{bmatrix} \end{array}, \tag{2.14}$$

The update is made, $\hat{S} \leftarrow \hat{S} + S_{\{i,j\}}$, and the process is repeated for next offending pair in the updated $\hat{S}$. Then, the *lumped coarse-grid matrix* is used as the coarse-grid operator,

$$\hat{A}_c \leftarrow \hat{S} - G, \tag{2.15}$$

instead of $A_c = RAP$. The positivity of η guarantees that no new zeros are introduced, thus preserving irreducibility in $\hat{A}_c$. Note that this process does not change the left kernel of A_c and, at convergence of the multilevel method, the right kernel is unaltered as well.

3 Recombination framework

Assume we have some version of Algorithm 1 that produces a sequence of iterates, $\{\mathbf{x}_i\}_{i=1}^{\infty}$, designed to approximate the solution of problem (1.1). At the k-th iteration, let the last m iterates be columns of an $n \times m$ matrix,

$$X = [\mathbf{x}_k,\ \mathbf{x}_{k-1},\ \ldots,\ \mathbf{x}_{k-m+2},\ \mathbf{x}_{k-m+1}], \tag{3.1}$$

with $\mathbf{x}_k$ being the newest, or best, iterate. We call m the *window size*. All columns of X are assumed to have the following properties:

$$\mathbf{x}_i > 0 \quad \text{and} \quad \|\mathbf{x}_i\|_1 = 1, \qquad i = 1, ..., n. \tag{3.2}$$

The natural question arises: is there a linear combination of these m iterates that is optimal in some sense? If the method that produces iterates $\{\mathbf{x}_i\}_{i=1}^{\infty}$ is a stationary, preconditioned residual correction, such as the weighted-Jacobi iteration or a fixed and additive multigrid correction, the standard answer to this question is to use a Krylov acceleration technique. The approaches in [14, 16], however, are nonlinear update schemes, where the multigrid hierarchy is changing with each iteration. Nevertheless, we take a fairly standard type of approach, similar to the approach given in [23] applied to FAS on nonlinear PDE problems. Both approaches are essentially generalized versions of Krylov acceleration that attempt to minimize the (nonlinear) residual of a linear combination of iterates, each modified for their respective problems.

We define the subset of *probability vectors* in n-dimensional space to be

$$\mathcal{P} := \{\mathbf{w} \in \mathbb{R}^n \quad \text{such that} \quad \|\mathbf{w}\|_1 = 1, \ \text{and} \quad \mathbf{w} \geq 0\}. \tag{3.3}$$

The framework requires a functional $\mathcal{F}(\mathbf{w})$ that is uniquely minimal in $\mathcal{P}$ at the solution to (1.1). The aim is to minimize this functional within a subset, $\mathcal{V} \subset \mathcal{R}(X)$, with the additional constraint equations $\|\mathbf{w}\|_1 = 1$ and $\mathbf{w} \geq 0$, which are used to ensure that $\mathbf{w}$ is a probability vector. Formally, this is

$$\text{minimize} \ \ \mathcal{F}(\mathbf{w}) \ \ \text{within} \ \ \mathcal{V} := \mathcal{P} \cap \mathcal{R}(X) \tag{3.4}$$

We label the constraints imposed on set $\mathcal{V}$ in the following way:

(C1) (Normalization Constraint) $\|\mathbf{w}\|_1 = 1$
(C2) (Nonnegativity Constraints) $\mathbf{w} \geq 0$
(C3) (Subspace Constraint) $\mathbf{w} \in \mathcal{R}(X)$

Note that **(C1)** is a single *equality constraint* while **(C2)** is a set of *inequality constraints*. Also, **(C3)** is technically a set of equality constraints which determine a linear subspace of $\mathbb{R}^n$:

$$\text{for } i = 1, \ldots, n-m, \quad \langle \mathbf{y}_i, \mathbf{w} \rangle = 0 \quad \text{where} \quad \text{span}\{\mathbf{y}_1, \ldots, \mathbf{y}_{n-m}\} = \mathcal{R}(X)^{\perp}.$$

However, because $m << n$ and $\dim(\mathcal{R}(X)^{\perp}) \approx n$, it is more convenient (and equivalent) to use the fact that there exists a vector $\mathbf{z}$ such that $\mathbf{w} = X\mathbf{z}$ for any $\mathbf{w}$ satisfying **(C3)**. This approach is preferred versus explicitly addressing the constraint equations, which are less accessible and inefficient to deal with.

The target functional, $\mathcal{F}(\mathbf{w})$, must be designed to have certain properties on the constrained subset: (i) the probability vector from $\mathcal{P}$ that minimizes $\mathcal{F}$ is unique and the solution to (1.1); and (ii) it is possible to approximate the minimizing vector within $\mathcal{P}$ in an efficient way. Due to the significance of the one-norm in the application, one expects that a functional involving this norm is ideal. Functionals involving the one-norm easily address property (i), but using the one-norm causes difficulty for property (ii), due to the non-differentiability of the functional. Instead, the squared two-norm is exploited

to address both of these properties. The following result shows that property (i) is upheld by using two standard functionals involving the squared two norm. Discussion in Sections 3 and 4 addresses how these functionals also address property (ii).

Theorem 3.1 (Functional Minimization) *A vector* $\mathbf{x} \in \mathcal{P}$ *attains the minimum in both*

$$\mathcal{F}_1(\mathbf{w}) := \frac{\langle A\mathbf{w},\ A\mathbf{w}\rangle}{\langle \mathbf{w},\ \mathbf{w}\rangle} \quad \textit{and} \quad \mathcal{F}_2(\mathbf{w}) := \langle A\mathbf{w},\ A\mathbf{w}\rangle\,, \tag{3.5}$$

if and only if $\mathbf{x}$ *is the steady-state solution to Eq.* 1.2.

Proof Clearly, $\mathcal{F}_1(\mathbf{w})$ and $\mathcal{F}_2(\mathbf{w})$ are greater to or equal to zero, $\forall \mathbf{w} \in \mathcal{P}$, with zero given only by $\mathbf{w} \in$ null(A). By the Perron-Frobenius theorem there is a unique null vector of A such that $\mathbf{x} > 0$ and $\|\mathbf{x}\|_1 = 1$. □

Remark 3.1 The choice of applying the minimization to solve problem, (1.2), is critical. For example, it will not work to attempt to maximize $\langle B\mathbf{w}, B\mathbf{w}\rangle$, as the maximizing vector in $\mathcal{P}$ is the right singular vector corresponding to the maximal singular value of B, which is not necessarily $\mathbf{x}$. Consider the simple example matrix

$$B = \frac{1}{4}\begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}.$$

For this example, the steady-state solution is $\mathbf{x} = [2/3,\ 1/3]^t$, but the direction that maximizes $\langle B\mathbf{w}, B\mathbf{w}\rangle$ is given by the normalized maximal singular vector, $\mathbf{w} \approx [0.53,\ 0.47]^t$. However, the steady-state solution is a right eigenvector of A with eigenvalue 0, so it is necessarily a right singular vector with singular value 0 as well.

The following two sections present two different acceleration approaches: the first employs unconstrained minimization of $\mathcal{F}_1$ within subspace $\mathcal{R}(X)$ and the second employs constrained minimization of $\mathcal{F}_2$ within constrained space $\mathcal{V}$.

4 Unconstrained minimization approach

The first approach we consider is to ignore constraints **(C1)** and **(C2)** and minimize $\mathcal{F}_1$ within $\mathcal{R}(X)$. That is, we pick any vector, $\mathbf{x}^*$, such that

$$\mathbf{x}^* = \underset{\mathbf{w}\in\mathcal{R}(X)}{\operatorname{argmin}} \ \frac{\langle A\mathbf{w},\, A\mathbf{w}\rangle}{\langle \mathbf{w},\, \mathbf{w}\rangle}. \tag{4.1}$$

Then, we check if $\mathbf{x}^*$ violates the positivity constraint, **(C2)**. If so, we perform a *backup*, meaning we decrease the window size by redefining X to contain the last $m-1$ iterates, and then repeat the minimization of $\mathcal{F}_1$ within the

smaller subspace. This process is repeated until $\mathbf{x}^*$ satisfies **(C2)**. Lastly, we enforce **(C1)** by normalizing in the one-norm, $\mathbf{x}^* \leftarrow \mathbf{x}^*/\|\mathbf{x}^*\|_1$. The details of this unconstrained minimization approach are presented in this section.

Remark 4.1 This process is guaranteed to eventually satisfy **(C2)** because when $m = 1$, the optimal vector is merely set to $\mathbf{x}_k$, which is a probability vector. The process of backing up is further explained in Section 4.1. Using this unconstrained approach, we assume that $\mathbf{x}^*$ is very unlikely to violate **(C2)** and the validity of this assumption is reinforced by many numerical tests where these violations were monitored. For problems where backup is more frequent, the constrained minimization approach presented in Section 5 is a better approach.

Remark 4.2 Normalization constraint **(C1)** is a scaling, and $\mathcal{F}_1$ is indifferent to scalings:

$$\frac{\langle A(\alpha\mathbf{w}),\ A(\alpha\mathbf{w})\rangle}{\langle(\alpha\mathbf{w}),\ (\alpha\mathbf{w})\rangle} = \frac{\langle A\mathbf{w},\ A\mathbf{w}\rangle}{\langle\mathbf{w},\ \mathbf{w}\rangle}, \quad \forall\mathbf{w} \neq \mathbf{0}, \quad \forall\alpha \in \mathbb{R} \setminus \{0\}.$$

Solving (4.1) without **(C1)** and normalizing afterward produces the same solution (with less computation) as solving with **(C1)** explicitly enforced.

The minimization problem (4.1) is solved by choosing a vector

$$\mathbf{x}^* = X\mathbf{z} = z_1\mathbf{x}_k + z_2\mathbf{x}_{k-1} + \ldots + z_m\mathbf{x}_{k-m+1}. \tag{4.2}$$

where coefficients $\mathbf{z}$ are selected to be any solution to a smaller minimization problem,

$$\mathbf{z} = \underset{\mathbf{v}\neq\mathbf{0}}{\operatorname{argmin}} \frac{\langle AX\mathbf{v},\ AX\mathbf{v}\rangle}{\langle X\mathbf{v},\ X\mathbf{v}\rangle}, \tag{4.3}$$

In other words, $\mathbf{z}$ is a right eigenvector of $(X^tX)^{-1}(X^tA^tAX)$ corresponding to the eigenvalue of smallest magnitude. Note that this is an $m \times m$ eigensystem with real and nonnegative spectrum. In exact arithmetic, solving (4.3) for such a $\mathbf{z}$, and setting $\mathbf{x}^* = X\mathbf{z}$, gives the optimal approximation in $\mathcal{R}(X)$. This is an eigenvector problem of order m which, for small m, is solved with a small amount of computation, relative to the per-iteration cost of the method being accelerated. For small window sizes, $m = 2, 3,$ or 4, this method of computing $\mathbf{x}^*$ is typically adequate and is the method used in the numerical results section.

For larger window sizes, the numerical stability of $(X^tX)^{-1}(X^tA^tAX)$ is a potential problem, and the accuracy of $\mathbf{z}$ may suffer. To avoid this pitfall, we consider finding orthogonal representations of matrices X and AX to form a more numerically stable problem of order m. First, apply QR factorization to the *input space* involved in the denominator of (4.3), $\mathcal{R}(X)$, and the *output space* involved in the numerator, $\mathcal{R}(AX) = \mathcal{R}(AQ_{in})$.

$$X = Q_{in}R_{in}, \qquad AQ_{in} = Q_{out}R_{out}. \tag{4.4}$$

Note that the QR factorization of AX is known as well without computing a third factorization:

$$AX = AQ_{in}R_{in} = Q_{out}R_{out}R_{in} = Q_{out}(R_{out}R_{in}). \tag{4.5}$$

These factorizations give us an equivalent problem that is better behaved in terms of numerical stability. By the QR factorization of X, for any vector $\mathbf{s} \in \mathcal{R}(X)$, there is a set of coefficients $\mathbf{u}$ such that $\mathbf{s} = Q_{in}\mathbf{u}$

$$\mathbf{s} = X\mathbf{v} = Q_{in}R_{in}\mathbf{v} = Q_{in}\mathbf{u}, \tag{4.6}$$

where $\mathbf{u} = R_{in}\mathbf{v}$. Using this fact, the QR factorization of AQ_{in}, and the orthogonality of Q_{in} and Q_{out} gives an equivalent minimization functional:

$$\begin{aligned}\frac{\langle AX\mathbf{v},\ AX\mathbf{v}\rangle}{\langle X\mathbf{v},\ X\mathbf{v}\rangle} &= \frac{\langle AQ_{in}\mathbf{u},\ AQ_{in}\mathbf{u}\rangle}{\langle Q_{in}\mathbf{u},\ Q_{in}\mathbf{u}\rangle} \\ &= \frac{\langle Q_{out}R_{out}\mathbf{u},\ Q_{out}R_{out}\mathbf{u}\rangle}{\langle Q_{in}\mathbf{u},\ Q_{in}\mathbf{u}\rangle} \\ &= \frac{\langle R_{out}\mathbf{u},\ R_{out}\mathbf{u}\rangle}{\langle \mathbf{u},\ \mathbf{u}\rangle}\end{aligned}$$

Any minimizer, $\mathbf{y}$, of this functional is a right singular vector of R_{out}, corresponding to its smallest singular value. Thus, $\mathbf{x}^* = Q_{in}\mathbf{y}$ is a minimizer of $\mathcal{F}_1$ in $\mathcal{R}(X)$.

Remark 4.3 The relative sizes of the diagonal entries of R_{in} indicate how linearly independent the columns of X are. This information could be used to adjust the window size adaptively, however, this has not been addressed in this work.

4.1 Backing up

If $\mathbf{x}^*$ violates **(C2)**, then using it to form a coarse grid within the algebraic multilevel method causes the coarsening algorithms to break down. The presence of vanishing or negative components in iterates $\mathbf{x}_k$ destroys the singular M-matrix nature of operators $A\,\mathrm{diag}(\mathbf{x}_k)$, such that the existence of unique positive solutions to the singular equations is no longer guaranteed. If $\mathbf{x}^*$ violates **(C2)**, we *back up* the acceleration by only using the last $m-1$ iterates to form a new optimal vector. This process is repeated until we have an optimal vector that satisfies **(C2)**, which is guaranteed. If $m = 1$, the subspace is merely the span of the last iterate, $X = \mathbf{x}_k$, which is output from Algorithm 1 and is necessarily a probability vector. We call this scenario a *full backup*, which amounts to no acceleration of the method with the additional overhead computational cost. The results in Section 6 show this scenario is unlikely for many example problems.

The rest of this section describes the details of the process used to backup the window size. For $p \leq m$, define the matrix of the last p iterates

$$X^{(p)} = \left[\mathbf{x}_k, \mathbf{x}_{k-1}, \ldots, \mathbf{x}_{k-p+1}\right]. \tag{4.7}$$

and solve a $p \times p$ unconstrained minimization problem,

$$\mathbf{x}_p^* = \underset{\mathbf{w}\in\mathcal{R}(X^{(p)})}{\operatorname{argmin}} \frac{\langle A\mathbf{w}, A\mathbf{w}\rangle}{\langle \mathbf{w}, \mathbf{w}\rangle}. \tag{4.8}$$

Note that $\mathbf{x}_m^* = \mathbf{x}^*$ and $X^{(m)} = X$. To solve for $\mathbf{x}_p^*$, we need to form matrices $(X^{(p)})^t X^{(p)}$ and $(X^{(p)})^t A^t A X^{(p)}$, find $\mathbf{z}_p$, the minimal right eigenvector of

$$[(X^{(p)})^t X^{(p)}]^{-1}[(X^{(p)})^t A^t A X^{(p)}]$$

and set $\mathbf{x}_p^* = X^{(p)}\mathbf{z}_p$. The entries of these matrices are

$$[(X^{(p)})^t X^{(p)}]_{ij} = \left\langle \mathbf{x}_{k-i-1}, \mathbf{x}_{k-j-1}\right\rangle \tag{4.9}$$

and

$$[(X^{(p)})^t A^t A X^{(p)}]_{ij} = \left\langle A\mathbf{x}_{k-i-1}, A\mathbf{x}_{k-j-1}\right\rangle. \tag{4.10}$$

These matrices are computed the first time only (when $p = m$), stored, and when $p < m$, they are reused. Therefore, the cost of backing up is a $p \times p$ eigenvector solve which is considered irrelevant to the $\mathcal{O}(n)$ method. The situation is similar when using the alternative approach that involves the QR factorizations, which is useful for calculating the minimizing vector with a larger window size.

4.2 Overhead cost estimates

Finding the minimizing vector in the range of X requires an eigenvector solve involving $(X^t X)^{-1}(X^t A^t A X)$, and computing the matrices $X^t X$ and $X^t A^t A X$ requires several inner products of order n. Computing $X^t X$ with $n \times m$ matrix X requires computing $m(m+1)/2$ inner products,

$$\left\langle \mathbf{x}_i, \mathbf{x}_j\right\rangle \quad \text{for} \quad k \geq i \geq j \geq k - m + 1. \tag{4.11}$$

On the next iteration, however, we can recycle the inner products from the previous iteration. Only m inner products will be new. They are

$$\left\langle \mathbf{x}_{k+1}, \mathbf{x}_j\right\rangle \quad \text{for} \quad j = k+1, k-1, \ldots, k-m+2 \tag{4.12}$$

The situation is the same for computing $X^t A^t A X$; only m inner products will be new. Therefore, assuming $k \geq m$, there are $2m$ inner product computations and one residual evaluation required per acceleration step.

Algorithm 3: Acceleration by Unconstrained Minimization
$\mathbf{x} \leftarrow$**AUM**$(A, \mathbf{x}_0^*, \tau, \mathrm{M})$

0. Set $k = 1$, if no initial guess is provided, choose $\mathbf{x}_0^*$.
1. Run the multilevel method,

$$\mathbf{x}_k \leftarrow \mathbf{AMMM}(A_1, \mathbf{x}_{k-1}^*, \nu_1, \nu_2, \gamma)$$

2. Set $m \leftarrow \min\{M, k\}$. `/* set window size */`
3. Set $X \leftarrow [\mathbf{x}_k, \mathbf{x}_{k-1}, ..., \mathbf{x}_{k-m+2}, \mathbf{x}_{k-m+1}]$. `/* last m iterates */`
4. Solve

$$\mathbf{y} = \underset{\mathbf{w} \in \mathcal{R}(X)}{\operatorname{argmin}} \frac{\langle A\mathbf{w}, A\mathbf{w} \rangle}{\langle \mathbf{w}, \mathbf{w} \rangle}$$

`/* if (C2) is not satisfied, backup and solve again */`
5. **if** $\mathbf{y} > 0$ **then** $\mathbf{x}_k^* = \frac{1}{\|\mathbf{y}\|_1}\mathbf{y}$, **else** set $m \leftarrow m - 1$ and go to 3.
6. Check for convergence, $\|A\mathbf{x}_k^*\|_1 < \tau$. Otherwise set $k \leftarrow k + 1$, and go to 1.

5 Constrained minimization approach

The second approach we consider is to minimize $\mathcal{F}_2$ in $\mathcal{R}(X)$ with both **(C1)** and **(C2)** explicitly enforced.

$$\mathbf{x}^* = \underset{\mathbf{w} \in \mathcal{R}(X) \cap \mathcal{P}}{\operatorname{argmin}} \langle A\mathbf{w}, A\mathbf{w} \rangle \tag{5.1}$$

If **(C2)** holds, then the absolute values in $\|\mathbf{w}\|_1$ are unnecessary. Thus, **(C1)** is a linear constraint, $\sum_i w_i = 1$. Furthermore, because $\mathbf{w} \in \mathcal{R}(X)$, there exists a vector $\mathbf{z}$ such that $\mathbf{w} = X\mathbf{z}$. This implies that

$$\|\mathbf{w}\|_1 = \sum_i^n w_i = \sum_{i=1}^n \sum_{j=1}^m X_{ij} z_j = \sum_{j=1}^m z_j \sum_{i=1}^n X_{ij} = \sum_{j=1}^m z_j, \tag{5.2}$$

due to each column in X being a probability vector. Therefore, the constrained subset is equivalently written as

$$\mathcal{R}(X) \cap \mathcal{P} = \left\{ \mathbf{w} = X\mathbf{z} \,:\, \sum_{i=1}^m z_i = 1\,,\ X\mathbf{z} \geq \mathbf{0} \right\}. \tag{5.3}$$

This is a *convex subset* of $\mathbb{R}^m$ defined by a single equality constraint and a large number, n, of inequality constraints. Formally, we rewrite (5.1) as

$$\begin{aligned} \text{minimize:} \quad & \mathbf{z}^t (X^t A^t A X)\mathbf{z}, \\ \text{subject to:} \quad & \mathbf{1}^t \mathbf{z} = 1, \quad \text{and} \\ & X\mathbf{z} \geq \mathbf{0}. \end{aligned} \tag{5.4}$$

A solution to (5.1) is given by any vector

$$\mathbf{x}^* = X\mathbf{z} = z_1 \mathbf{x}_k + z_2 \mathbf{x}_{k-1} + ... + z_m \mathbf{x}_{k-m+1}, \tag{5.5}$$

where coefficients z_i are selected to minimize $\langle AX\mathbf{z}, AX\mathbf{z}\rangle$ with the equality constraint satisfied, $\sum_{j=1}^{m} z_j = 1$, and the full set of inequality constraints satisfied, $\sum_{j=1}^{m} x_{ij} z_j \geq 0$, for any $1 \leq i \leq n$.

For the $m = 2$ case, we are guaranteed that only two constraints are necessary, and the other $n - 2$ constraints may be ignored when solving (5.1). This is explained and displayed in Fig. 3, but the algebraic details are given in [17]. For slightly larger window sizes, $m = 3$ or 4, we assume that only a few of these constraints are relevant and that the constrained minimization is performed in $\mathcal{O}(n)$ operations. The implementation for this paper uses the active set method from matlab's quadprog function [8].

If any of the inequality constraints are *active*, or equivalently, if $(\mathbf{x}^*)_i = 0$ for any i, there are potential difficulties for Algorithm 1. The coarsening

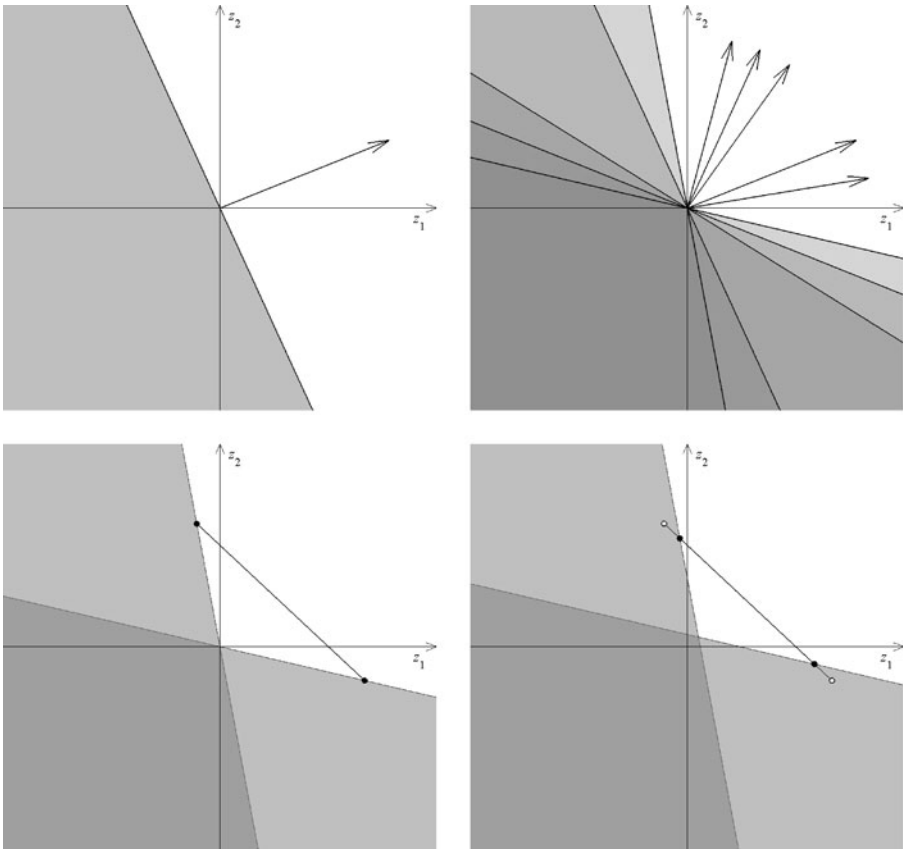

Fig. 3 Constrained minimization with window size $m = 2$. The *top-left* shows how a single inequality constraint in **(C2)** limits (z_1, z_2). The *shaded regions* are infeasible. The *top-right* shows that the intersection of subsets satisfying each constraint is the region satisfying the two most extreme constraints. The *line segment in the bottom-left* shows the location of the subset satisfying the equality constraint. The *bottom right* shows the feasibility region for $\delta > 0$

procedures involved in aggregation and AMG need to assume that the input is an all-positive vector. Otherwise, there are columns of all zeros in $A\,\mathrm{diag}(\mathbf{x}_k)$, so the M-matrix structure is not upheld. Essentially, the well-posedness of the algorithm is lost when an entry in the input vector is allowed to be non-positive. There are two ways to avoid using a vector with a zero entry in the coarsening. The first is to minimize over an *interior* subset, $\mathcal{R}(X) \cap \mathcal{P}_\delta$, with

$$\mathcal{P}_\delta := \{\mathbf{w} \in \mathbb{R}^n \quad \text{such that} \quad \|\mathbf{w}\|_1 = 1, \ \text{and} \quad \mathbf{w} \geq \delta\, x_{min}\}, \tag{5.6}$$

where δ is a small positive number ($\delta = 0.1$, for example) and x_{min} is the smallest entry in X.

The other way to avoid a zero component is to allow the pre-relaxation of the next cycle to make the iterate strictly positive. Often enough, one single relaxation will enforce **(C2)** in this case, but it may be necessary to do more. The following two results show that the solution to the constrained minimization problem (5.1) will have the property $\mathbf{w} > 0$ after some relaxation steps.

Theorem 5.1 (Pre-Relaxation Positivity) *Assume that A is an irreducible singular M-matrix and that weighted-Jacobi relaxation parameter α is in* $(0, 1)$. *If vector $\mathbf{w} \geq 0$ and $\mathbf{w} \not\equiv \mathbf{0}$ in any neighborhood within the graph of A, then the relaxed vector is positive, $(I - \alpha D^{-1} A)\mathbf{w} > 0$.*

Proof Matrix A is a singular M-matrix, so for any $i \neq j$ we have $a_{ij} \leq 0$. There is also at least one negative off-diagonal entry in every row of A, since it is irreducible. Define N_i to be the neighborhood of i in the graph of A, excluding i. Then

$$0 > \sum_{j \in N_i} a_{ij} w_j \qquad \text{for any } i. \tag{5.7}$$

Because $\alpha \in (0, 1)$, $a_{ii} > 0$ and $w_i \geq 0$, we have

$$(1 - \alpha) a_{ii} w_i > \alpha \sum_{j \in N_i} a_{ij} w_j, \quad \forall i \tag{5.8}$$

This implies

$$(1 - \alpha) w_i - \alpha \frac{1}{a_{ii}} \sum_{j \in N_i} a_{ij} w_j > 0, \quad \forall i \tag{5.9}$$

which is the same as $(I - \alpha D^{-1} A)\mathbf{w} > 0$. □

The following corollary is a generalization of the previous theorem that can be easily proved. It shows that there is some amount of pre-relaxation that guarantees positivity.

Corollary 5.2 *Assume that A is an irreducible singular M-matrix and that weighted-Jacobi relaxation parameter α is in* $(0, 1)$. *For any vector $\mathbf{w} \geq 0$ that*

is not identically zero, there exists an integer $\nu > 0$ *such that* $\mathbf{w} \not\equiv \mathbf{0}$ *in any neighborhood within the graph of* $(I - \alpha D^{-1} A)^{\nu}$. *Therefore,* $(I - \alpha D^{-1} A)^{\nu} \mathbf{w} > 0$.

The existence of such an integer ν that meets the assumptions of the previous corollary is certain (set ν to the diameter of the graph), but for a general $\mathbf{w} \geq \mathbf{0}$ this integer could be unacceptably high. In practice, however, small ν (one or two) is sufficient for all the problems we have investigated. This is because the constrained minimization is unlikely to return a vector that is identically zero on any large localized patches within the graph of A.

6 Numerical results

In this section, we present the results of applying the unconstrained (Algorithm 3) and constrained acceleration (Algorithm 4) approaches with window sizes $m = 1, 2, 3, 4$ to versions of Algorithm 1 for several examples. Here, the accelerators are applied to V-cycles ($\gamma = 1$) for the SAM [16] and AMG [14] versions of Algorithm 1 and W-cycles ($\gamma = 2$) for unsmoothed aggregation [15] and "smooth P only" SA versions, as defined in (2.8).

For all examples, the specific set of parameters in this paragraph are used. One pre- and post-relaxation step is used at each stage of the algorithm and $\gamma = 1$ or 2 (V(1,1) or W(1,1)-cycles). The iterative method used for relaxation is weighted Jacobi with relaxation parameter $\alpha = 0.7$. Direct coarse-level solves are performed using the techniques from [14–16]. The lumping parameter is $\eta = 0.01$. Initial guesses $\mathbf{x}^{(0)}$ are randomly sampled in $(0, 1)$ and normalized to one in the one norm.

For the examples involving aggregated multigrid hierarchies, the neighborhood-based aggregation technique from [21] is used, as discuss in Section 2.1, with strength-of-connection defined as in (2.6) with $\theta = 0.25$.

Algorithm 4: Acceleration by Constrained Minimization
$\mathbf{x} \leftarrow$**ACM**$(A, \mathbf{x}_0^*, \tau, \mathrm{M}, \delta)$

0. Set $k = 1$, if no initial guess is provided, choose $\mathbf{x}_0^*$.
1. Run the multilevel method,

$$\mathbf{x}_k \leftarrow \mathbf{AMMM}(A_1, \mathbf{x}_{k-1}^*, \nu_1, \nu_2, \gamma)$$

2. Set $m \leftarrow \min\{M, k\}$ `/* set window size */`
3. Set $X \leftarrow [\mathbf{x}_k, \mathbf{x}_{k-1}, ..., \mathbf{x}_{k-m+2}, \mathbf{x}_{k-m+1}]$. `/* last m iterates */`
4. Define $\mathcal{P}_\delta := \{\mathbf{w} \in \mathbb{R}^n \quad \text{such that} \quad \|\mathbf{w}\|_1 = 1, \ \text{and} \quad \mathbf{w} \geq \delta x_{min}\}$.
5. Solve

$$\mathbf{x}_k^* = \underset{\mathbf{w} \in \mathcal{R}(X) \cap \mathcal{P}_\delta}{\operatorname{argmin}} \langle A\mathbf{w}, A\mathbf{w} \rangle \tag{5.10}$$

6. **if** $\|A\mathbf{x}_k^*\|_1 > \|A\mathbf{x}_k\|_1$ **then** $\mathbf{x}_k^* \leftarrow \mathbf{x}_k$
7. Check convergence, $\|A\mathbf{x}_k^*\|_1 < \tau$. Otherwise set $k \leftarrow k + 1$, and go to 1.

Smoothing parameters (α_R, α_P) were chosen to be $(0, 0)$ when using unsmoothed aggregation, $(0, 0.7)$ when smoothing P only, and $(0.7, 0.7)$ when smoothing R and P.

For the examples involving multigrid hierarchies employing standard AMG, strength-of-connection is defined by (2.9), with $\theta = 0.25$.

The parameter $\delta = 0.1$ is used to maintain positivity when defining constraints (5.6) in the constrained minimization approach. No explicit study regarding sensitivity to the parameters (δ, η) was done, mainly due to the success of the initial choices.

The following statistics are reported in tables throughout the rest of this section. The *number of levels* in the multigrid hierarchies is denoted by "lvls". The *iteration count*, "its", is the lowest integer K such that $\|A\mathbf{x}^{(K)}\|_1 / \|A\mathbf{x}^{(0)}\|_1 < 10^{-8}$. The *operator complexity*, C_{op}, is the total number of nonzero entries in the problem matrices, A_l, from every level in the multigrid hierarchy relative to the number of nonzero entries in A. This number is an estimate for the amount of work performed by the relaxation processes on all levels. The amount of lumping required within each multigrid hierarchy is not reported here, but is reported in [14, 16] for common examples.

6.1 Example problems

Example 6.1 (2d Lattice) We consider a Markov chain on a 2d lattice with uniform weights. Matrix A is essentially a scaled graph-Laplacian on a 2D uniform quadrilateral lattice with 5-point stencil. (see Fig. 4, or [14, 16] for more complete descriptions). The results of accelerating SA and AMG versions of Algorithm 1 by unconstrained and constrained wrappers with small window sizes are reported in Table 1.

For both types of acceleration, similar results are observed with window size 3 giving the most effective acceleration. Window size 4 takes more overhead

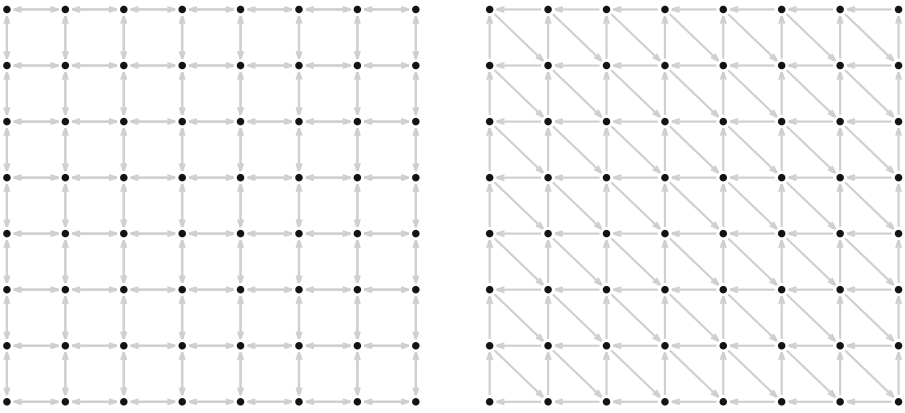

Fig. 4 Graphs for Examples 6.1 and 6.4. *Black nodes* represent states within the Markov chain and *gray lines* represent transitions where *arrows* specify directionality

Table 1 Example 6.1 (2d lattice)

n	lvls	C_{op}	its	Unconstrained			Constrained		
				Window size			Window size		
				2	3	4	2	3	4
SAM									
64	3	1.26	16	12	9	9	11	10	10
256	3	1.34	17	12	9	9	12	10	9
1,024	4	1.32	17	13	10	10	11	11	11
4,096	5	1.34	18	13	11	11	13	11	11
16,384	5	1.33	18	13	10	10	12	11	11
65,536	6	1.34	19	14	11	11	13	12	11
MCAMG									
64	3	2.02	11	9	7	8	9	7	7
256	5	2.20	11	9	8	8	8	8	8
1,024	6	2.20	11	9	8	8	9	8	7
4,096	7	2.20	11	9	8	8	9	8	8
16,384	8	2.20	11	9	8	8	9	8	8
65,536	9	2.20	11	9	8	8	9	8	8

Iteration counts for various window sizes for unconstrained and constrained minimization strategies applied to SAM and MCAMG methods. The iteration count for the standalone versions of these methods is below the column labeled "its". Additionally, number of levels and operator complexities of the multigrid hierarchies used are given

and typically offers little or no improvement over window size 3. For the SA method, iteration counts are reduced by around 40% and for the AMG method, iteration counts are reduced by around 30%. No backup steps are needed for unconstrained minimization to maintain iterate positivity.

Example 6.2 (Random Planar Graph) We consider Markov chains based on unstructured, random, planar graphs (see [16]). To construct the transition matrix for the chain, we start by randomly distributing n nodes in $(0, 1)^2$. Then we form a planar graph connecting these nodes using Delaunay triangulation and put bidirectional links connecting each node that shares an edge within the triangulation. The probability of transitioning from node i to node j is given by the reciprocal of the number of outward links from node i (a random walk). The acceleration wrappers perform similarly for this example in comparison to Example 6.1 with SA iteration counts reduced by around 60% and AMG iteration counts reduced by around 35% (Table 2).

Example 6.3 (Random Planar Graph, Nonsymmetric) We use the unstructured planar graphs from the previous example to form a similar problem, but with nonsymmetric sparsity structure. Starting with the graphs described in Example 6.2, we select a subset of triangles from the triangulation such that no two triangles in the set share an edge. This is done by selecting any triangle, marking it with a "+", and marking all of its three neighbors with a "−". This process is repeated for the next unmarked triangle until all triangles are marked. One edge on each "+" triangle is next made uni-directional by randomly deleting one of the six directed arcs that connect the three nodes in

Table 2 Example 6.2 (random planar graph)

n	lvls	C_{op}	its	Unconstrained			Constrained		
				Window size			Window size		
				2	3	4	2	3	4
SAM									
1,024	4	1.29	25	16	12	12	17	14	13
2,048	4	1.29	29	18	13	13	18	15	14
4,096	4	1.32	32	21	14	14	19	17	15
8,192	5	1.34	28	19	14	13	17	16	14
16,384	5	1.34	39	25	14	14	19	16	15
32,768	5	1.35	39	26	16	16	21	18	17
MCAMG									
1,024	6	2.13	15	11	10	9	11	10	9
2,048	7	2.22	14	10	9	9	10	9	9
4,096	7	2.19	15	11	10	9	11	10	9
8,192	8	2.25	15	11	10	10	11	10	10
16,384	8	2.26	15	11	10	10	11	10	10
32,768	9	2.28	14	11	10	10	11	10	10

See Table 1 for full description

the triangle. Note that this makes some of the "−" triangles have missing arcs as well. In fact, the "−" may have several missing directed arcs, but each "+" triangle has one and only one missing directed arc. This process ensures that the resulting Markov chain is still irreducible. The probability of transitioning from node i to node j is given by the reciprocal of the number of outward links from node i. See Fig. 5 for a small version of this example with the "+" triangles marked. The quality of the acceleration is identical to the previous example, as seen in Table 3.

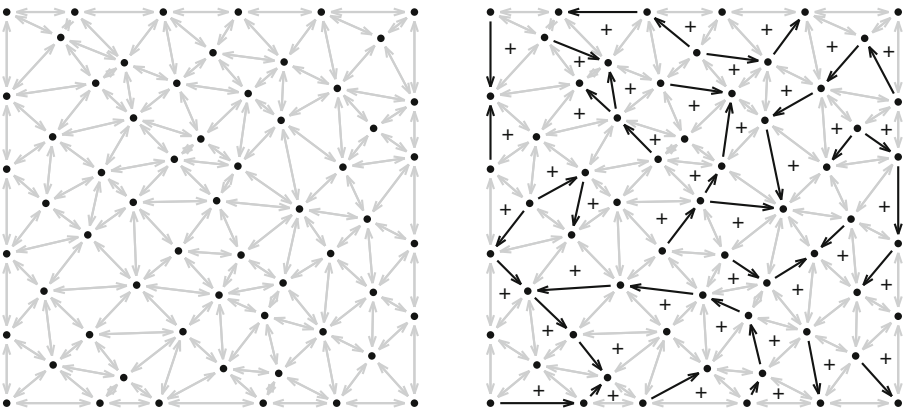

Fig. 5 Graphs for small versions of Examples 6.2 (*left*) and 6.3 (*right*). *Black dots* represent nodes, and *light gray arrows* represent bidirectional links. For the *figure on the right*, *black arrows* represent uni-directional links and triangles with a "+" inside have a single link that was made uni-directional. For easier visualization, the *graphs* shown here have a more regular distribution of points than the actual points used to build the Markov chains

Table 3 Example 6.3 (random planar graph, nonsymmetric)

n	lvls	C_{op}	its	Unconstrained			Constrained		
				Window size			Window size		
				2	3	4	2	3	4
SAM									
1,024	4	1.31	39	24	15	15	20	17	17
2,048	4	1.31	29	20	15	14	17	17	16
4,096	4	1.35	69	25	22	21	28	20	18
8,192	4	1.37	35	23	15	15	18	17	16
16,384	5	1.36	42	28	17	16	21	19	18
32,768	5	1.38	44	28	17	16	21	17	18
MCAMG									
1,024	6	2.67	14	10	10	10	11	10	9
2,048	7	2.62	15	11	10	10	11	10	10
4,096	8	2.70	16	12	11	10	12	11	10
8,192	8	2.74	16	12	11	10	12	11	10
16,384	9	2.77	16	12	11	11	12	11	11
32,768	10	2.79	17	12	11	11	12	11	11

Table 1 has a complete description

Figure 6 displays convergence histories for the SAM method applied to Example 6.3 with unconstrained acceleration on the left and constrained acceleration on the right, each with window sizes m =1,2, and 3. The histories for $m = 4$ were very similar to $m = 3$ and were therefore not displayed.

Example 6.4 (Tandem Queueing Network) We consider the Markov chain given by two serial queues of finite capacity with the following transition probabilities: the probability of a new customer entering the system is 0.32, the probability of a customer being processed by the first queue and moving to the second queue is 0.36, and the probability of a customer being processed by the second queue and leaving the system is 0.32. The graph of this Markov

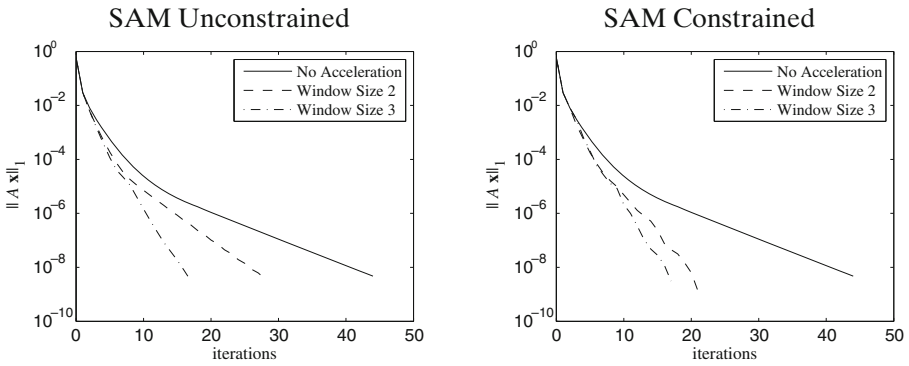

Fig. 6 Convergence histories for SAM with unconstrained and constrained minimization with various window sizes for Example 6.3 and $n = 32{,}768$

Table 4 Example 6.4 (tandem queueing network)

n	lvls	C_{op}	its	Unconstrained			Constrained		
				Window size			Window size		
				2	3	4	2	3	4
SAM									
256	3	1.25	17	15	15	15	15	15	15
1,024	4	1.25	20	17	17	16	17	16	16
4,096	4	1.24	19	17	16	16	17	16	16
16,384	5	1.24	22	18	18	17	19	18	16
65,536	6	1.25	18	17[a]	17[a]	16[a]	17	17	16
MCAMG									
256	5	4.08	15	12	11	11	12	10	10
1,024	7	4.39	15	12	11	11	12	11	11
4,096	8	4.47	15	13	13	12	13	12	12
16,384	9	4.54	15	14	14	13	15	14	14
65,536	12	4.60	16	16	15	14[a]	16	14	15

Table 1 has a complete description
[a]Cases where backup steps are performed

chain is planar and is represented by directed edges on a triangulation of the unit square (see Fig. 4 and [16] for a more complete description).

For both types of acceleration, similar results are observed in Table 4, where for both SAM and MCAMG, the low iteration counts are not significantly improved. However, results for accelerating a less successful standalone method (SAM with smoothing P only and not smoothing R) are given in Table 5. For this method, both types of acceleration give similar improvement, where about 65% less iterations are needed for the largest problem size. The acceleration wrappers reduce the iteration counts significantly, and the accelerated methods are much more near optimal that the unaccelerated. It should be noted that the accelerator applied to SAM with smoothing P only is still not as efficient as the standalone version of SAM with smoothing both R and P. These results are meant to display how the acceleration typically improves nonoptimal methods, thus increasing robustness for multiplicative algebraic

Table 5 Example 6.4 (tandem queueing network)

n	lvls	C_{op}	its	Unconstrained			Constrained		
				Window size			Window size		
				2	3	4	2	3	4
SAM (Smooth P only)									
256	3	1.24	32	20	17	16	20	17	16
1,024	4	1.22	41	27	20	20	27	20	20
4,096	4	1.23	56	37	24	24	35	24	24
16,384	5	1.22	57	37	27	27	38	26	26
65,536	6	1.22	80	39	28	28	36	28	31

Table 1 has a complete description

multilevel methods. Additionally, for problems where smoothing R and P gives unacceptable operator complexities, using SAM with smoothing P only and an acceleration wrapper may prove more efficient.

The results in both Tables 4 and 5 show that a small amount of backup steps were required for certain problem sizes and window sizes. For the largest problem size, $n = 65{,}536$, and window size $m = 4$, a few backup steps are needed for unconstrained minimization to maintain iterate positivity. For MCAMG, the window size is reduced to $m = 3$ for 2 of the 14 iterations. No full backups were observed.

Example 6.5 (Petri Net) We consider a stochastic Petri net (SPN) problem. Petri nets are a formalism for the description of concurrency and synchronization in distributed systems. They consist of: places, which model conditions or objects; tokens, which represent the specific value of the condition or object; transitions, which model activities that change the value of conditions or objects; and arcs, which specify interconnection between places and transitions. A stochastic Petri net is a standard Petri net, together with a tuple $\Lambda_s = (r_1, ..., r_s)$ of exponentially distributed transition firing rates. A finite place, finite transition, marked stochastic Petri net is isomorphic to a discrete space Markov process. See [10] for an in-depth discussion of Petri nets.

Again, both types of acceleration produced similar results, as seen in the top part of Table 6. For SAM, the low iteration counts are not really improved. Again, results for accelerating a less successful standalone method (unsmoothed aggregation and W-cycles) are given in the bottom part of Table 6. For this method, both types of acceleration give similar improvement, where about 50% less iterations are needed for the largest problem size. The acceleration is slightly better for the smaller versions of this problem. It should

Table 6 Example 6.5 (stochastic Petri net)

n	lvls	C_{op}	its	Unconstrained			Constrained		
				Window size			Window size		
				2	3	4	2	3	4
SAM									
819	4	1.85	16	13	13	12	13	13	12
2,470	4	1.93	14	14	12	12	13	12	12
10,416	5	2.05	14	14	13	12	14	12	12
23,821	5	2.04	15	15	13	13	15	13	13
45,526	5	1.90	14	14	13	14	14	13	13
Unsmoothed aggregation with W-cycles									
819	4	1.79	61	32	25	24	29	26	23
2,470	5	1.85	63	31	27	25	30	30	28
10,416	6	1.90	62	33	28	31	33	33	33
23,821	6	1.92	62	34	32	24	32	34	34
45,526	6	1.94	63	39	38	32	37	36	35

Table 1 has a complete description

be noted that the accelerator applied to unsmoothed aggregation and W-cycles is still not as efficient as the standalone version of SAM with smoothing both R and P and V-cycles. The acceleration techniques, however, are shown to increase the robustness of algebraic multiplicative methods, in general.

7 Conclusion

In this work we developed two approaches to accelerate adaptive algebraic multiplicative multilevel methods for steady-state solution to Markov chains. One acceleration approach is based on minimizing a quadratic rational functional in an unconstrained subspace, and the other is based on minimizing a quadratic functional in a constrained subset. The unconstrained minimization technique offers a strategy that has $\mathcal{O}(n)$ cost for all window sizes. The cost of the constrained minimization technique with window size two is also $\mathcal{O}(n)$, whereas using window sizes greater than two has the possibility of greater cost. Although we have not quantified the probability of needing greater cost, we assume this need is unlikely due to our observations within the class of problems presented here. We performed tests by applying the accelerators to two different classes of adaptive algebraic multiplicative multilevel methods, one based on aggregation and one based on algebraic multigrid. For both the unconstrained and constrained approaches, similar results were observed. In some cases where the standalone methods were performing optimally, reductions to iteration counts were observed. However, for a few cases where the unaccelerated methods were already near optimal, the accelerated methods offered no improvement. Significant improvements in iteration counts and scalability could be made with small window sizes when the standalone methods were not performing optimally. Therefore, the accelerators were found to be useful to increase the robustness of a given method with a small amount of additional cost, similar to the effect of preconditioned Krylov acceleration applied to nonsingular linear problems.

References

1. Berman, A., Plemmons R.J.: Nonnegative Matrices in the Mathematical Sciences. SIAM, Philadelphia (1987)
2. Brandt, A.: Multi-level adaptive solutions to boundary-value problems. Math. Comput. **31**(138), 333–390 (1977)
3. Brandt, A.: Algebraic multigrid theory: the symmetric case. Appl. Math. Comput. **9**, 23–26 (1986)
4. Brandt, A., McCormick, S., Ruge, J.: Algebraic multigrid (AMG) for sparse matrix equations. In: Evans, D.J. (ed.) Sparsity and its Applications, pp. 257–284. Cambridge University Press, Cambridge (1984)
5. Brezina, M., Falgout, R., MacLachlan, S., Manteuffel, T., McCormick, S., Ruge, J.: Adaptive smoothed aggregation (αsa) multigrid. SIAM Rev. (SIGEST) **47**, 317–346 (2005)
6. Brezina, M., Falgout, R., Maclachlan, S., Manteuffel, T., McCormick, S., Ruge, J.: Adaptive algebraic multigrid. SIAM J. Sci. Comput. (SISC) **27**, 1261–1286 (2006)

7. Briggs, W., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial, 2nd edn. SIAM Books, Philadelphia (2000)
8. Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic, London (1981)
9. Horton, G., Leutenegger, S.T.: A Multi-level Solution Algorithm for Steady-state Markov Chains, pp. 191–200. ACM SIGMETRICS (1994)
10. Molloy, M.K.: Performance analysis using stochastic Petri nets. IEEE Trans. Comput. **C**(31), 913–917 (1982)
11. Saad, Y.: Iterative Methods for Sparse Linear Systems. SIAM Books, Philadelphia (2003)
12. Saad, Y., Schultz, M.H.: GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems. SIAM J. Sci. Comput. (SISC) **7**(3), 856–869 (1986)
13. Schweitzer, P.J., Kindle, K.W.: An iterative aggregation–disaggregation algorithm for solving linear equations. Appl. Math. Comput. **18**, 313–354 (1986)
14. De Sterck, H., Manteuffel, T.A., McCormick, S.F., Miller, K., Ruge, J., Sanders, G.: Algebraic multigrid for Markov chains. SIAM J. Sci. Comput. **32**, 544–562 (2010)
15. De Sterck, H., Manteuffel, T.A., McCormick, S.F., Nguyen, Q., Ruge, J.: Multilevel adaptive aggregation for Markov chains, with application to web ranking. SIAM J. Sci. Comput. (SISC) **30**, 2235–2262 (2008)
16. De Sterck, H., Manteuffel, T.A., McCormick, S.F., Pearson, J., Ruge, J., Sanders, G.: Smoothed aggregation multigrid for Markov chains. SIAM J. Sci. Comput. (SISC) **32**, 40–61 (2009)
17. De Sterck, H., Miller, K., Sanders, G., Winlaw, M.: Recursively accelerated multilevel aggregation for Markov chains. SIAM J. Sci. Comput. **32**, 1652 (2010)
18. Stewart, W.J.: An Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton (1994)
19. Treister, E., Yavneh, I.: Square and stretch multigrid for stochastic matrix eigenproblems. Numer. Linear Algebra Appl. **17**, 229–251 (2009)
20. Trottenberg, U., Osterlee, C.W., Schuller, A.: Multigrid (Appendix A: An Introduction to Algebraic Multigrid) (Appendix by K. Stuben). Academic, New York (2000)
21. Vaněk, P., Mandel, J., Brezina, M.: Algebraic multigrid on unstructured meshes. Technical Report X, Center for Computational Mathematics, Mathematics Department (1994)
22. Virnik, E.: An algebraic multigrid preconditioner for a class of singular m-matrices. SIAM J. Sci. Comput. (SISC) **29**(5), 1982–1991 (2007)
23. Washio, T., Oosterlee, C.W.: Krylov subspace acceleration for nonlinear multigrid schemes. Electron. Trans. Numer. Anal. **6**, 271–290 (1997)