Fast multilevel methods for Markov chains

Hans De Sterck[1,*] Killian Miller[1], Eran Treister[2] and Irad Yavneh[2]

[1] *Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada (hdesterck@uwaterloo.ca, k7miller@uwaterloo.ca).*
[2] *Department of Computer Science, Technion - Israel Institute of Technology, Haifa 32000, Israel (eran@cs.technion.ac.il, irad@cs.technion.ac.il).*

SUMMARY

This paper describes multilevel methods for the calculation of the stationary probability vector of large, sparse, irreducible Markov chains. In particular, several recently proposed significant improvements to the multilevel aggregation method of Horton and Leutenegger are described and compared. Furthermore, we propose a very simple improvement of that method using an over-correction mechanism. We also compare with more traditional iterative methods for Markov chains such as weighted Jacobi, two-level aggregation/disaggregation, and preconditioned stabilized biconjugate gradient and GMRES. Numerical experiments confirm that our improvements lead to significant speedup, and result in multilevel methods that are competitive with leading iterative solvers for Markov chains. Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: Markov chain, stationary probability vector, multigrid, multilevel aggregation, over-correction

1. Introduction

This paper describes improvements to the multilevel aggregation method of Horton and Leutenegger [1] for Markov chains. The main contribution of this paper is an automatic over-correction mechanism that can cheaply and effectively improve the convergence of the basic multilevel aggregation method presented in [1]. We further demonstrate how the recently developed on-the-fly (OTF) adaptive approach [2], which uses interlaced multiplicative and additive correction cycles to compute the stationary distribution, can significantly improve the convergence of standalone multilevel solvers for Markov chains. In particular we accelerate the multilevel aggregation method of [1], and the Markov chain algebraic multigrid (MCAMG) method of [3] using the OTF approach. We also compare with more traditional iterative methods for Markov chains such as weighted Jacobi, two-level aggregation/disaggregation, and preconditioned stabilized biconjugate gradient (Bi-CGStab) and GMRES. Numerical results are obtained for a variety of test problems including a tandem queueing problem [4], a random

*Correspondence to: k7miller@uwaterloo.ca walk on an unstructured directed planar graph [5], and a multi-class, finite buffer priority system [6, 7]. The parameters in the multi-class, finite buffer priority system are chosen in such a way that we obtain a nearly completely decomposable (NCD) Markov chain.

Horton and Leutenegger were among the first to consider numerical methods for Markov chains with more than two levels [1, 8], see also [9]. Their method is a direct extension of the two-level iterative aggregation/disaggregation method for Markov chains due to Takahashi [10], which makes use of the aggregated equations proposed by Simon and Ando [11]. Since the pioneering work of Takahashi, two-level methods for Markov chains have been considered widely in the Markov chain literature [12, 13, 14, 15, 16, 17, 18, 19]. These methods have been shown to be particularly effective for NCD Markov chains, where the known structure of the problem can be exploited by the aggregation process, which typically results in fast convergence. Domain decomposition methods [20, 21], and projection methods such as Arnoldi, GMRES, Lanczos, conjugate gradient and biconjugate gradient [6, 22, 4], have also been shown to be effective at solving for the stationary distribution of Markov chains. These methods constitute some of today's leading solvers for Markov chains. Historically, multilevel methods have been largely disregarded for Markov chain problems, however, recent work has shown promise for such methods. While theoretical convergence results are very difficult to obtain, especially for general problems with nonsymmetric sparsity structure, in many cases empirical evidence has demonstrated good convergence properties and robustness of multilevel methods for Markov chains [23, 24, 25, 3, 5, 26, 27, 2]. We note that while theoretical convergence results do exist for certain classes of two-level methods [15, 13, 17, 18, 19], these results typically deal with only local convergence and do not extend easily (if at all) to multilevel methods. In fact, theoretical results for AMG solvers applied to nonsymmetric problems are quite rare, with advances having only recently been made (see [28] and the references therein). Moreover, as far as we know, there is no convergence theory for any type of multilevel adaptive method related to what we present in this paper.

In this paper we consider irreducible homogeneous Markov chains with a finite state space. For a discrete time Markov chain (DTMC) with transition probability matrix given by an $n \times n$ column-stochastic matrix B, the problem of finding the stationary probability vector $\mathbf{x}$ may be stated as follows. We seek the unique vector $\mathbf{x}$ such that

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

In this paper we follow the notational convention, customary in the numerical linear algebra literature on eigenvalue problems, that all vectors (including probability vectors) are column vectors and that all transition matrices are column-oriented. We find it beneficial to work with an equivalent formulation of (1.1) in terms of the singular M-matrix generated by B. Mathematically, we seek the vector $\mathbf{x} \in \mathbb{R}^n$ such that

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

where $A = I - B$ is an irreducible singular M-matrix.

Since B is a stochastic matrix, its spectral radius $\rho(B) = 1$ and $\lambda_1 = 1$ is a dominant eigenvalue of B. Given that B is nonnegative and irreducible with $\rho(B) = 1$, the Perron-Frobenius theorem [29] guarantees the existence of a unique solution to (1.1). Moreover, the solution has strictly positive components. A *subdominant eigenvalue* λ_2 of B satisfies

$$|\lambda_2| = \max\{|\lambda| : |\lambda| < 1,\, \lambda \in \Lambda(B)\}$$

where $\Lambda(B)$ is the spectrum of B. We are interested in Markov chains for which $|\lambda_2| \to 1$ as the size of the state space increases. Traditional iterative methods for computing the stationary probability vector may be unacceptably slow to converge for such problems due to poor damping of the error component associated with the subdominant eigenvalue [6, 4]. Markov chains with this property are said to be *slowly mixing*. Multilevel iterative methods aim to accelerate convergence for this type of problem by reducing error components with different scales on progressively coarser levels. These methods are attractive because they have the potential to be *asymptotically optimal*, i.e., they achieve approximate solutions up to a given tolerance with a convergence factor per iteration that is bounded above by a constant $c < 1$, resulting in a computational effort that grows linearly with the problem size [30].

The rest of this paper is organized as follows. In Section 2 we briefly describe the basic multilevel aggregation framework of Horton and Leutenegger for Markov chains. In Section 3 we describe a number of different ways in which the basic multilevel aggregation framework for Markov chains can be accelerated. These include the recently proposed Markov chain AMG (MCAMG) method and the on-the-fly adaptive framework, and a new automatic over-correction routine. In Section 4 we present numerical results comparing new and existing multilevel approaches with other existing iterative methods, and in Section 5 we make our closing remarks.

2. Multilevel aggregation for Markov chains

In this section we describe the basic two-level aggregation algorithm from which our multilevel methods are derived. For further details we refer the reader to [1, 24, 25]. Notationally, objects that pertain to the current level have no subscript, and those that pertain to the next coarser level have the subscript c. The symbols representing transfer operators have neither subscripts nor superscripts.

We are interested in solving the fine-level problem:

$$A\,\mathbf{x} = \mathbf{0} \tag{2.1}$$

where $A = I - B$. We can rewrite the exact solution, $\mathbf{x}$, in terms of the current approximation, $\mathbf{x}_i$, and its unknown multiplicative error, $\mathbf{e}_i$, as $\mathbf{x} = \operatorname{diag}(\mathbf{x}_i)\,\mathbf{e}_i$. Here $\operatorname{diag}(\mathbf{x}_i)$ is a diagonal matrix with $\mathbf{x}_i$ on the diagonal. Substituting into (2.1) we obtain an equivalent fine-level problem:

$$A\operatorname{diag}(\mathbf{x}_i)\,\mathbf{e}_i = \mathbf{0}. \tag{2.2}$$

Note that we have to assume here that all components of the current approximation, $\mathbf{x}_i$, are strictly positive. At convergence, the multiplicative error is known, $\mathbf{e}_i = \mathbf{1}$, where $\mathbf{1}$ is the vector of all ones.

Our goal is to obtain a coarse-level representation of (2.2). We proceed by aggregating the n fine-level degrees of freedom into $n_c < n$ groups according to the columns of the aggregation matrix $Q \in \mathbb{R}^{n \times n_c}$, where $q_{ij} = 1$ if fine-level node i belongs to aggregate j and $q_{ij} = 0$ otherwise. In a multilevel setting, the aggregation matrix Q can be determined at successive coarse levels using topological information when the Markov chain is structured [1]. For example, in a two-level setting, iterative aggregation/disaggregation (IAD) methods applied to nearly completely decomposable (NCD) Markov chains [10, 15, 16, 4] exploit the known block structure of the transition probability matrix when building the coarse aggregated system. Alternatively, aggregates can be chosen algebraically based on *strength of connection* in the problem matrix [24, 8, 26].

Given the aggregation matrix Q, the coarse-level operator, A_c, is defined by the Galerkin operator

$$A_c := Q^T A \operatorname{diag}(\mathbf{x}_i) Q = RAP \tag{2.3}$$

where $R = Q^T$ is the restriction operator and $P = \operatorname{diag}(\mathbf{x}_i)Q$ is the interpolation operator. An important property of the interpolation operator P, is that it should have the exact solution $\mathbf{x}$ approximately in its range, i.e., $\mathbf{x} \approx P\,\mathbf{t}_c$ for some coarse vector $\mathbf{t}_c$. By noting that $\mathbf{x}_i = P\mathbf{1}_c$, we observe that as the fine-level approximation improves, i.e., as $\mathbf{x}_i \to \mathbf{x}$, so too does the approximation of the exact solution in the range of P. Additionally, as the approximate solution becomes more accurate, the coarse representation of the original problem will also improve. Because A is typically nonsymmetric for Markov chains, following [31], we also require that R^T should have the *near-nullspace* of A^T (i.e., vectors $\mathbf{e}$ such that $A^T\mathbf{e} \approx \mathbf{0}$) in its range. Since the left nullspace is known for Markov chains (it is the vector of all ones), this condition can be satisfied exactly, i.e., $R^T\mathbf{1}_c = Q\mathbf{1}_c = \mathbf{1}$.

Given the definition of the coarse operator in (2.3), the coarse-level problem can be stated as

$$A_c\,\mathbf{e}_c = \mathbf{0}. \tag{2.4}$$

Using $\mathbf{e}_c$ one can define an improved coarse-level approximation, $\mathbf{x}_c := \operatorname{diag}(R\,\mathbf{x}_i)\,\mathbf{e}_c$, and (2.4) can be modified to obtain the alternative coarse-level formulation:

$$A_c \operatorname{diag}(R\,\mathbf{x}_i)^{-1}\,\mathbf{x}_c = \mathbf{0}. \tag{2.5}$$

In the case of direct, non-overlapping aggregation, (2.5) has a straightforward probabilistic interpretation [10, 1, 4]. Once (2.4) has been solved for $\mathbf{e}_c$, we can obtain the next iterate, $\mathbf{x}_{i+1}$, via the *coarse-level correction* formula:

$$\mathbf{x}_{i+1} = P\,\mathbf{e}_c. \tag{2.6}$$

Note that one may also solve (2.5) for $\mathbf{x}_c$, which leads to the equivalent coarse-level correction formula

$$\mathbf{x}_{i+1} = P\ \operatorname{diag}(R\,\mathbf{x}_i)^{-1}\,\mathbf{x}_c. \tag{2.7}$$

In this paper we use the weighted Jacobi method for all relaxation operations. At each level ν_1 pre-relaxations and ν_2 post-relaxations are performed, except on the coarsest level where a direct solve is performed. One iteration of weighted Jacobi applied to the general problem $A\,\mathbf{x} = \mathbf{b}$ is given by

$$\mathbf{x}_{new} = \mathbf{x} + \omega D^{-1}(\mathbf{b} - A\,\mathbf{x}) = \operatorname{Relax}(A, \mathbf{x}, \mathbf{b}) \tag{2.8}$$

where D is the diagonal part of A and ω is the relaxation parameter. We note that given strict positivity of the elements of $\mathbf{x}$, one can show that the relaxed approximation $(I - \omega D^{-1}A)\mathbf{x}$ (in the case of $\mathbf{b} = \mathbf{0}$) also has strictly positive elements provided that $\omega \in (0, 1]$.

Instead of aggregating only once and solving the coarse-level problem (2.4) directly, we can obtain a multilevel method by applying the two-level method recursively. A high-level description of the resulting multilevel method for Markov chains is given by Algorithm 1. Solution of the coarsest-level subproblem via the Grassmann, Taksar, Heyman (GTH) algorithm [32, 33] can be justified by observing that A is an irreducible singular M-matrix on all levels. Assuming that the initial fine-level approximation has strictly positive elements, this property of the coarse-level operators further implies that the coarse-grid corrected approximations $\tilde{\mathbf{x}}$ must also be strictly positive on all levels. We note that the parameter μ in Algorithm 1 determines the cycle type; for $\mu = 1$ we obtain V-cycles, and for $\mu = 2$ we obtain W-cycles. Note finally that the approach of Algorithm 1, originally presented in [1], can also be interpreted as a precursor of classes of methods for the solution of linear systems that have become known as adaptive algebraic multigrid (AMG) or smoothed aggregation (SA) [34, 35] or exact interpolation schemes [27].

Algorithm 1: Multilevel aggregation for Markov chains $\mathbf{x} \leftarrow \mathrm{AGG}(\mathrm{A}, \mathbf{x}, \mu, \nu_1, \nu_2)$

```
Input: Operator: $A \in \mathbb{R}^{n\times n}$, strictly positive vector: $\mathbf{x} \in \mathbb{R}^n$, cycle index: $\mu$,
       number of pre-relaxations: $\nu_1$, number of post-relaxations: $\nu_2$.
Output: New approximation to the solution of $A\,\mathbf{x} = \mathbf{0}$
if not at coarsest level then
    1. $\hat{\mathbf{x}} \leftarrow \mathrm{Relax}(A, \mathbf{x}, \mathbf{0})$ $\nu_1$ times
    2. Build $Q$ based on $A$ and $\hat{\mathbf{x}}$
    3. Set $R \leftarrow Q^T$, $P \leftarrow \mathrm{diag}(\hat{\mathbf{x}})Q$
    4. Set $\mathbf{x}_c \leftarrow R\,\hat{\mathbf{x}}$, and repeat $\mu \geq 1$ times:
           $\mathbf{x}_c \leftarrow \mathrm{AGG}(A_c(\mathrm{diag}(R\,\hat{\mathbf{x}}))^{-1},\ \mathbf{x}_c,\ \mu,\ \nu_1,\ \nu_2)$
    5. Coarse-grid correction (CGC): $\tilde{\mathbf{x}} \leftarrow P(\mathrm{diag}(R\,\hat{\mathbf{x}}))^{-1}\mathbf{x}_c$
    6. $\mathbf{x} \leftarrow \mathrm{Relax}(A, \tilde{\mathbf{x}}, \mathbf{0})$ $\nu_2$ times
else
    7. Direct solve of $A\,\mathbf{x} = \mathbf{0}$, $\mathbf{1}^T\mathbf{x} = 1$ by the GTH algorithm
end
```

3. Acceleration of multilevel aggregation

In this section we describe different ways in which the pure multilevel aggregation algorithm described in Section 2 can be accelerated. Numerical results comparing the approaches will be given in Section 4. Our main contribution is a new automatic over-correction procedure, which as demonstrated in the section on numerical results is an inexpensive way to accelerate basic multilevel aggregation for Markov chains. We also consider the recently developed on-the-fly (OTF) adaptive approach [2], which uses interlaced multiplicative and additive correction cycles to obtain the stationary distribution. The OTF approach is illustrated in conjunction with the Markov chain algebraic multigrid (MCAMG) algorithm from [3], which is a recently developed improvement of Algorithm 1, and is described briefly in Section 3.1. In the interest of space, we do not consider other recent approaches for accelerating Algorithm 1, which include smoothed aggregation multigrid for Markov chains [25], recursively accelerated multilevel aggregation for Markov chains [5, 26], and square and stretch multigrid for Markov chains [27].

3.1. Markov Chain Algebraic Multigrid

In this section we briefly describe the Markov chain algebraic multigrid algorithm developed in [3]. While the MCAMG algorithm does not use aggregation to construct the coarse-level problem, its structure is similar to that of Algorithm 1. Instead of aggregation, the MCAMG algorithm employs an algebraic multigrid coarsening procedure to define the coarse grid and build the transfer operators. The construction of the transfer operators proceeds in two phases. In the first phase, the current fine-level is partitioned into sets of coarse points C and fine points F, where the C-points determine the coarse-level degrees of freedom. This is accomplished by a two-pass coarsening routine, which attempts to satisfy two complementary heuristics that strive to limit the number of C-points, while at the same time produce a coarse level with good approximation properties. Coarsening relies on the notion of *strength of connection* in the scaled fine-level matrix $A\,\mathrm{diag}(\mathbf{x}_i)$, which quantifies the relative importance of one variable in determining the value of another, depending on the magnitude of the elements in $A\,\mathrm{diag}(\mathbf{x}_i)$. The strength of connection for the MCAMG algorithm is defined as follows. Given a threshold value $\theta \in [0, 1]$, point j *strongly influences* point i if

$$-a_{ij}x_j \geq \theta \max_{k\neq i}\{-a_{ik}x_k\}. \tag{3.1}$$

In this paper, unless stated otherwise, we use strength threshold $\theta = 0.25$. We note that this definition of strength of connection is directional, i.e., when we say that j is strongly connected to i, then j strongly influences i, or i strongly influences j, but not necessarily both. This is in contrast to other similar definitions of strength of connection which are inherently symmetric, e.g., see [25, 5]. For further information regarding AMG strength of connection and the two-pass coarsening procedure we refer to [30].

Once the sets C and F are known, the interpolation operator P can be constructed. Interpolation is accomplished by approximating the error at each F-point as a weighted sum of the error at C-points. Without restricting generality, suppose that the current set of fine-level points, $H = \{1, \ldots, n\}$, is ordered so that

$$H = \{\underbrace{1, \ldots, n_c}_{C}, \underbrace{n_c + 1, \ldots, n_c + n_f}_{F}\},$$

where $|C| = n_c$, $|F| = n_f$ and $n = n_c + n_f$. Then, for any point $i \in H$, we require that

$$(P\,\mathbf{e}_c)_i = \begin{cases} (\mathbf{e}_c)_i & \text{if } i \in C, \\ \sum_{j\in C_i} w_{ij}(\mathbf{e}_c)_j & \text{if } i \in F, \end{cases} \tag{3.2}$$

where $\mathbf{e}_c$ is the coarse-level error approximation, w_{ij} are the interpolation weights, and C_i is the set of C-points that strongly influence point i according to (3.1). For Markov chains the weights are defined in such a way that P has nonnegative entries with unit row sums. The sparsity structure of P is similar to the sparsity structure of the smoothed aggregation multigrid interpolation operator with overlapping aggregates [25], however, their approximation properties clearly differ. For further details regarding the computation of the interpolation weights and the properties of the interpolation operator we refer to [3].

Once P has been constructed we define the coarse level operator according to $A_c = RA\bar{P}$, where the restriction operator $R = P^T$ and $\bar{P} = \text{diag}(\mathbf{x}_i)P$. Due to the structure of R and $\bar{P}$ the coarse-level operator may not be an irreducible singular M-matrix. Since this is essential to our algorithm, we make use of a *lumping* technique, which produces a lumped coarse-level operator $\hat{A}_c$ with the desired structure by adding symmetric perturbations to A_c. Given that the lumped coarse-level operators are irreducible singular M-matrices on all levels, one can show that the coarse-grid corrected iterates must be strictly positive on all levels. In particular, the new finest-level approximation must be strictly positive (recall that positivity is preserved by the relaxation operations). For a detailed explanation of the lumping technique we refer to Section 4.3 in [3]. The remainder of the MCAMG algorithm is essentially the same as the multilevel aggregation algorithm described in Section 2, with the coarse-level problem defined in terms of the lumped coarse-level operator $\hat{A}_c\,\mathbf{e}_c = \mathbf{0}$, and a coarse-level correction given by $\mathbf{x}_{i+1} = \bar{P}\,\mathbf{e}_c$. Apart from these differences, the framework of the MCAMG algorithm is the same as that given by Algorithm 1.

3.2. On-the-fly adaptive setting

In this paper we apply the on-the-fly adaptive multigrid setting, described in detail in [2]. The main idea of this approach is that multiplicative correction schemes such as Algorithm 1, are equivalent to the classical multigrid additive correction scheme, provided the two algorithms use the same adaptive multigrid operators that are updated in every step. The advantage of an equivalent additive correction formulation of Algorithm 1 is that the adaptive operators can be "frozen" after a few cycles, and those frozen additive cycles, for which the multigrid hierarchy of operators is not recalculated in each cycle, are considerably cheaper, without sacrificing much in terms of convergence speed.

Classical additive multigrid schemes for linear systems are based upon the following fundamental idea. Given the linear system

$$A\,\mathbf{x} = \mathbf{b}, \tag{3.3}$$

where $A \in \mathbb{R}^{n\times n}$ is a positive definite matrix, these methods apply a cheap, albeit slowly converging stationary iterative method, such as the weighted Jacobi method

$$\mathbf{x}_{new} = \mathbf{x} + \omega D^{-1}(\mathbf{b} - A\mathbf{x}) = \text{Relax}(A, \mathbf{x}, \mathbf{b}). \tag{3.4}$$

The slow convergence of these relaxations is usually due to a relatively small number of components in the error, referred to as *algebraically smooth*, which approximately satisfy the homogeneous system (2.1). To eliminate these error components, classical multigrid methods use an additive *coarse-grid correction* (CGC), applied by constructing and solving a coarse system with fewer degrees of freedom. Unlike Algorithm 1, the additive correction schemes approximate the error $\mathbf{e} = \mathbf{x} - \mathbf{x}_i$ on the coarse-grid, and not the exact solution $\mathbf{x}$ (or its multiplicative error). Algorithm 2 describes a typical two-level additive correction cycle. A multilevel V-cycle is obtained by recursively treating the coarse-grid problem in step 4. For Markov chains, this approach can be applied to a homogeneous system with $\mathbf{b} = \mathbf{0}$ on the finest grid. Of course, on coarser levels, the system $A_c\,\mathbf{e}_c = \mathbf{r}_c$ is inhomogeneous, with $\mathbf{r}_c \neq \mathbf{0}$. We note that on the coarsest level it is necessary to solve the linear system $A_c\,\mathbf{e}_c = \mathbf{r}_c$. Since the operator A_c may be singular, multiple solutions may exist. In this paper, as in [2], we use the pseudoinverse of A_c (i.e., $\mathbf{e}_c = A_c^{+}\,\mathbf{r}_c$) with a small drop tolerance to obtain a solution on the coarsest level. This results in a solution of $A_c\,\mathbf{e}_c = \mathbf{r}_c$ that does not contain any significant contribution of nullspace components of A_c, which if present may have contaminated the coarse-grid correction.

Algorithm 2: Two-level additive cycle

```
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: New approximation to the solution of $A\,\mathbf{x} = \mathbf{b}$
1. Apply pre-relaxations: $\mathbf{x} \leftarrow \mathrm{Relax}(A, \mathbf{x}, \mathbf{b})$.
2. Define the residual $\mathbf{r} \leftarrow \mathbf{b} - A\,\mathbf{x}$.
3. Restrict the residual: $\mathbf{r}_c \leftarrow R\,\mathbf{r}$.
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: $\mathbf{x} \leftarrow \mathrm{Relax}(A, \mathbf{x}, \mathbf{b})$.
```

Algorithm 2 requires the complete hierarchy of multigrid operators for all levels in advance. To obtain effective convergence, it is crucial that we choose the prolongation operator P such that the algebraically smooth components are approximately in its range. To achieve this for difficult problems, a complete framework of adaptive SA and AMG algorithms was developed for symmetric linear systems [34, 35]. These algorithms feature an additive solution phase, preceded by a multiplicative setup phase where the multigrid operators are constructed. The setup phase is targeted at solving the homogeneous system (2.1); it aims to find an approximation to the near-nullspace of each coarse-level operator, and then build the transfer operators accordingly. In practice, this is achieved by a multiplicative correction scheme algorithm, similar to Algorithm 1. Once the multigrid hierarchy of operators is set, the linear system is solved by the additive Algorithm 2, using the fixed operators from the setup phase.

It can easily be seen that multiplicative Algorithm 1 can equivalently be written in the form of additive Algorithm 2 as follows. For Algorithm 1, let the current approximation $\mathbf{x}_i$ be in the range of P, and let $\mathbf{e}_{c,i}$ be the coarse multiplicative error that satisfies $\mathbf{x}_i = P\,\mathbf{e}_{c,i}$. The coarse-grid equation of the multiplicative scheme is given by $RAP\,\mathbf{e}_c = \mathbf{0}$, with $\mathbf{e}_c$ the unknown multiplicative coarse-grid error, and the coarse-grid correction formula is $\mathbf{x}_{i+1} = P\,\mathbf{e}_c$. Now define the equivalent unknown additive coarse-grid error, $\hat{\mathbf{e}}_c$, by $\hat{\mathbf{e}}_c = \mathbf{e}_c - \mathbf{e}_{c,i}$. It is now easy to see that the coarse-grid correction formula can be written in additive form:

$$\mathbf{x}_{i+1} = P\,\mathbf{e}_c = P\,\hat{\mathbf{e}}_c + P\,\mathbf{e}_{c,i} = \mathbf{x}_i + P\,\hat{\mathbf{e}}_c,$$

and the coarse-grid equation of the equivalent additive formulation becomes

$$RAP\,\hat{\mathbf{e}}_c = RAP\,(\mathbf{e}_c - \mathbf{e}_{c,i}) = -RAP\,\mathbf{e}_{c,i} = -RA\,\mathbf{x}_i = R\,\mathbf{r}_i,$$

with the residual given by $\mathbf{r}_i = -A\,\mathbf{x}_i$. This shows that the multiplicative formulation of Algorithm 1 is equivalent to an additive formulation as in Algorithm 2, provided that the same operators are used for the two formulations, with P updated in every additive cycle such that the current approximation $\mathbf{x}_i$ is always in the range of P, and with $A_c = RAP$ updated in every cycle accordingly.

In the "on-the-fly" approach for Markov chains, we first perform a few multiplicative cycles (Algorithm 1), which from this point onward we also refer to as setup cycles. We then freeze the operators and perform additive cycles (Algorithm 2), which we also refer to as solution cycles. If the convergence speed of the additive cycles (which we measure by residual reduction) is not satisfactory, we switch back, on-the-fly, to one or more additional multiplicative cycles, which improves the operators. The underlying motivation for this approach is that solution cycles (with frozen operators) are considerably cheaper, but they require the operators supplied by the setup cycles, and better approximations supplied to the setup cycles normally yield better operators in return.

The goal of the on-the-fly algorithm is to first quickly and efficiently reach an initial tolerance $||A\mathbf{x}||_1 < \varepsilon_\alpha$, using solution cycles over setup cycles when possible. We try to minimize the number of expensive setup cycles by first performing just one setup cycle. Even though the operators produced by this single setup cycle may not be very accurate, they may suffice for obtaining useful solution cycles. As in [2], we then repeatedly apply the following procedure, where $V_{SOL}(\cdot)$ denotes a solution cycle, V_{SET} denotes a setup cycle, $q(\cdot)$ is some measure of the approximation error, and $0 < C \leq 1$ is a scalar threshold for an acceptable convergence factor of the solution cycles.

Procedure: try-SOL-else-SET(C)

1. $\mathbf{y} = V_{SOL}(\mathbf{x})$ (try a solution cycle)
2. If $q(\mathbf{y}) > q(\mathbf{x})$ do $\mathbf{x} \leftarrow V_{SET}(\mathbf{x})$ and return
   (if the solution cycle increases the error, do a setup cycle instead)
3. If $q(\mathbf{y}) < Cq(\mathbf{x})$ then $\mathbf{x} = \mathbf{y}$, else $\mathbf{x} \leftarrow V_{SET}(\mathbf{y})$
   (if the error reduction is better than the convergence factor threshold, accept **y**; if not, do an extra setup cycle)

For $q(\mathbf{x})$ we use

$$q(\mathbf{x}) = \frac{\|A\mathbf{x}\|_1}{\|\mathbf{x}\|_1}, \tag{3.5}$$

which is the L^1-norm of the residual. The residual, which emphasizes the high-energy modes, in general may not be an optimal measure of performance. However, we find it quite reliable in practice. Given the try-SOL-else-SET(C) procedure, the on-the-fly algorithm may now be stated as follows in Algorithm 3 below. We require that iterates produced by the try-SOL-else-SET(C) procedure, and in particular by the additive solution cycles be nonnegative. This is especially important when an iterate is used as input to a setup cycle. While the setup cycles have guaranteed positive solution components on all levels (see Sections 2 and 3.1), the solution cycles may admit negative values after the coarse-grid correction (with or without over-correction). Therefore, after each solution cycle we check that the updated approximation on the finest level has nonnegative elements, and if not, we take the absolute value and renormalize. We note that negative values typically only arise for very small components, in which case taking the absolute value and renormalizing does not influence the overall accuracy in any significant way. Effectively, we make a trade-off between the accuracy of small components and robustness.

```
Algorithm 3: On-the-fly adaptive MG
  Input: Initial tolerance: $\varepsilon_\alpha$, convergence threshold: $C$, operator: $A \in \mathbb{R}^{n\times n}$.
1. Initial Setup:
   Apply several relaxations on a strictly positive random initial guess $\mathbf{x}_0 \in \mathbb{R}^n$, $\mathbf{1}^T\mathbf{x}_0 = 1$,
   followed by a setup cycle to obtain $\mathbf{x}_1$.
2. Improve Solution Approximation:
   Repeat try-SOL-else-SET($C$), until $||A\mathbf{x}_i||_1 < \varepsilon_\alpha$.
3. Finalize Setup:
   Do another setup cycle so that $\mathbf{x}_i$ is in the range of $P$.
4. Solution:
   Apply solution cycles until the stopping criteria (see Section 4) are satisfied.
```

3.3. Multilevel aggregation with over-correction

The idea of applying over-correction is a simple one, its goal being to enhance the convergence of multigrid cycles based on aggregation [36, 37]. For additive cycles, instead of applying a standard CGC in step 5 of Algorithm 2, one may introduce an over-correction parameter α into the coarse-grid correction:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha P\mathbf{e}_c\,, \tag{3.6}$$

where α is typically a scalar larger than 1. This is motivated by the observation that while the correction typically approximates the error very well in the sense of its "progress", it may not provide a good approximation in the sense of its "size" [37]. Typically, a fixed value for α is used, which is chosen based on a trial-and-error strategy. The main challenge, however, is to determine an appropriate value for α automatically. In [37, 38, 39] a method for determining α was suggested by minimizing the A-norm of the error after the coarse-grid correction

$$||\mathbf{e}_{i+1}||_A^2 = \langle \mathbf{e}_{i+1}, A\mathbf{e}_{i+1}\rangle = ||\mathbf{e}_i + \alpha P\mathbf{e}_c||_A^2\,, \tag{3.7}$$

where $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ denote the unknown errors at iterations i and $i+1$, respectively. In practice, one needs to smooth the correction by applying a relaxation prior to computing α

$$\hat{\mathbf{e}} = \text{Relax}(A, P\mathbf{e}_c, \mathbf{0}). \tag{3.8}$$

The optimal α can then be calculated by

$$\alpha_{opt} = \frac{\hat{\mathbf{e}}^T(A\mathbf{x}_i - \mathbf{b})}{\hat{\mathbf{e}}^T A\hat{\mathbf{e}}} \tag{3.9}$$

Unfortunately, the A-norm is only suitable for SPD problems.

For a nonsymmetric operator A, expression (3.7) is no longer meaningful. As far as we can tell, only a fixed over-correction has been considered for nonsymmetric problems in the literature [40, 41, 42, 43]. Horton and Leutenegger also considered a fixed over-correction for Markov chains in [1], by taking a convex combination of two different coarse approximations to the correction. In order to automatically determine α, one may instead consider the residual L^2-norm as an alternative to the A-norm. In [44], the residual norm was used to accelerate the convergence of multilevel processes, but only as a top-level acceleration. Unfortunately, the residual norm may be a poor measure of performance, since it is much more sensitive to rough error modes introduced by multilevel over-correction than the A-norm. This is due to the fact that the residual norm is equivalent to the A^TA-norm. However, the residual norm is currently the best we have at our disposal. We attempt to make the residual norm less sensitive to rough modes by projecting the residual to the coarse level via the restriction operator. Restricting to the coarse level has a similar effect as smoothing, but is much cheaper than applying a relaxation. Following this line of reasoning we minimize the following functional with respect to α

$$||RA\mathbf{e}_{i+1}||_2^2 = ||RA(\mathbf{e}_i + \alpha\hat{\mathbf{e}})||_2^2, \tag{3.10}$$

where $\hat{\mathbf{e}}$ is described above in (3.8). It is easy to see that the global minimizer of (3.10) is given by

$$\alpha_{opt} = \frac{(RA\hat{\mathbf{e}})^T(R(A\mathbf{x}_i - \mathbf{b}))}{||RA\hat{\mathbf{e}}||_2^2}. \tag{3.11}$$

So far we have described the over-correction procedure used in the (additive) solution cycles, i.e., Algorithm 2. In the case of the multiplicative correction scheme, as in Algorithm 1, the CGC is given by

$$\tilde{\mathbf{x}} = P(\operatorname{diag}(R\,\hat{\mathbf{x}}))^{-1}\mathbf{x}_c, \tag{3.12}$$

where $\mathbf{x}_c$ is the solution of the coarse-grid problem and P is a prolongation that has the previous solution $\mathbf{x}_i$, exactly in its range (i.e., $\mathbf{x}_i = P\,\mathbf{t}_c$ for some coarse vector $\mathbf{t}_c$.) One way in which we can apply the over-correction technique to the multiplicative correction scheme, is to use a multiplicative over-correction of the form

$$\mathbf{x}_{i+1} \leftarrow X(X^{-1}\tilde{\mathbf{x}})^{\alpha}, \tag{3.13}$$

where $\alpha > 1$, $X = \operatorname{diag}(\mathbf{x}_i)$, and the αth power is applied componentwise. However, minimization of

$$||RAX(X^{-1}\tilde{\mathbf{x}})^{\alpha}||_2^2$$

with respect to α may be computationally expensive. As a workaround we use the fact that the correction achieved by our aggregation cycle is typically quite small, i.e., $\tilde{\mathbf{x}} = P\mathbf{e}_c = \mathbf{x}_i + X\boldsymbol{\varepsilon}$, where $\|\boldsymbol{\varepsilon}\|_\infty \ll 1$. Thus, it follows by Taylor expansion that

$$(X^{-1}\tilde{\mathbf{x}})^{\alpha} = (\mathbf{1} + \boldsymbol{\varepsilon})^{\alpha} \approx (\mathbf{1} + \alpha\boldsymbol{\varepsilon}) \tag{3.14}$$

and therefore, a linearized over-correction formula is given by

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + X\alpha\boldsymbol{\varepsilon} = (1-\alpha)\mathbf{x}_i + \alpha\tilde{\mathbf{x}}. \tag{3.15}$$

We note that this formula is equivalent to Equation (3.6) with $P\mathbf{e}_c = \tilde{\mathbf{x}} - \mathbf{x}_i$.

Similar to the additive correction scheme, in order to find the optimal parameter α, the corrected vector must first be smoothed:

$$\hat{\mathbf{x}} = \text{Relax}(A, \tilde{\mathbf{x}}, \mathbf{0}). \tag{3.16}$$

Then, the optimal parameter α is obtained by minimizing

$$||RA((1-\alpha)\mathbf{x}_i + \alpha\hat{\mathbf{x}})||_2^2, \tag{3.17}$$

where the minimizer is given by

$$\alpha_{opt} = \frac{(RA\mathbf{x}_i)^T RA(\mathbf{x}_i - \hat{\mathbf{x}})}{||RA(\hat{\mathbf{x}} - \mathbf{x}_i)||_2^2}. \tag{3.18}$$

In practice, α_{opt} is restricted to a predefined interval (in our test cases we use $[1.1, 2]$), since this results in a more stable algorithm, e.g., it is possible that Equation (3.18) can yield a negative value for α_{opt}, which may lead to numerical instability. If α_{opt} falls outside this interval, then it is set to either 1.1 or 2, depending on which boundary point it is nearest. Furthermore, it may be necessary to use a smoothing parameter in Equations (3.8) and (3.16) that is smaller than the parameter used in the pre- and post-relaxations, or, it may even be necessary to use more than one relaxation to smooth the coarse-grid correction. Insufficient smoothing of the coarse-grid correction may result in a poor determination of α_{opt}. Again, we note that if A were symmetric, then the optimal over-correction parameter should be based on a minimization of the A-norm instead of the restricted residual norm.

The over-correction procedure may be problematic for certain test cases when applied in conjunction with the on-the-fly framework, since amplifying the coarse-grid correction in (3.6) has a tendency to ruin strict positivity of the solution. This is most prevalent at the beginning of the solution process before the operators have been finalized. Hence, when using over-corrected pure aggregation in the on-the-fly framework, we accept the additive iterate in the try-SOL-else-SET(C) procedure only if its residual is considerably smaller than the residual of the previous iterate, otherwise we perform a setup cycle using the previous iterate as input. More precisely, we modify step 3 of the try-SOL-else-SET(C) procedure according to

$$\text{If } q(\mathbf{y}) < Cq(\mathbf{x}) \text{ then } \mathbf{x} = \mathbf{y}, \text{ else } \mathbf{x} \leftarrow V_{SET}(\mathbf{x})$$

with a C value that is bounded away from 1. We note that alternatively one could use solution cycles with less aggressive over-correction (or even without it) in the try-SOL-else-SET(C) procedure, and then only once the operators are finalized apply aggressive over-correction.

In the automatic approach, the over-correction parameter α is recalculated on each level. For comparison, we also include tests in which α is fixed for all levels and cycles to a nearly-optimal value obtained by trial and error. For the fixed α tests we use over-correction formulas (3.13) and (3.6) for the multiplicative and additive cycles, respectively.

4. Numerical results

In this section we present the results of our numerical tests for a variety of test problems. These include a tandem queueing problem, a multiclass, finite buffer priority system, and a random walk on an unstructured directed planar graph. Results were obtained for the weighted Jacobi method, two-level aggregation (the two-level version of Algorithm 1, similar to IAD), preconditioned stabilized biconjugate gradient (Bi-CGStab) [6], preconditioned GMRES [6], pure multilevel aggregation (AGG) (Algorithm 1, [1]), MCAMG [3], and pure multilevel aggregation with over-correction (OC-AGG). When referring to OC-AGG we are implicitly considering both the fixed and automatic approaches, unless a particular approach is specified. A prefix of OTF- before the name of a method indicates that it was executed using the on-the-fly framework described in Section 3.2. All methods were implemented in Matlab, making use of sparse and vectorized Matlab procedures as much as possible, and using external C routines for some of the bottleneck operations in the multilevel methods.

In this paper we propose stopping criteria based on the residual, however, other stopping criteria can be applied as well [45, 4]. We use the following stopping criterion for all methods except GMRES

$$\text{stop if} \quad k > maxit \quad \text{or} \quad \frac{\|A\,\mathbf{x}_k\|_1}{\|\mathbf{x}_k\|_1} < \tau \|A\,\mathbf{x}_0\|_1,$$

where $\tau > 0$ is the convergence tolerance, k is the iteration count and $maxit$ is the maximum number of iterations the algorithm will be allowed to perform. For GMRES we use

$$\text{stop if} \quad k > maxit \quad \text{or} \quad \Big(\|A\,\mathbf{x}_k\|_2 < \tau \|A\,\mathbf{x}_0\|_1 \quad \text{and} \quad \frac{\|A\,\mathbf{x}_k\|_1}{\|\mathbf{x}_k\|_1} < \tau \|A\,\mathbf{x}_0\|_1 \Big), \tag{4.1}$$

where "and" represents a logical AND operation with short-circuiting. The initial guess $\mathbf{x}_0$ is randomly generated with strictly positive elements and unit 1-norm, and the convergence tolerance is $\tau = 10^{-8}$. In order to check the one-norm criterion in (4.1) at each inner iteration of GMRES it is necessary to compute the current approximation $\mathbf{x}_k$. (In our efficient implementation $\mathbf{x}_k$ is not computed in the inner iterations, see [22].) However, this practice may incur unnecessary extra computations, especially when GMRES is still far from converging. Therefore, we use a condition based on the two-norm of the residual, which we get for free (see [22]), to gauge whether GMRES is near convergence, and only when this condition is satisfied do we compute $\mathbf{x}_k$ and check the one-norm criterion in (4.1). We note that although the iterates $\mathbf{x}_k$ are not normalized in the inner iterations, we have confirmed through empirical observation that their one-norms tend to remain close to one, and so the two-norm test is meaningful.

Weighted Jacobi results were obtained using an empirically determined optimal weight $\omega \in (0, 1]$ ($maxit =$ 20,000). Two-level aggregation results were obtained using two pre-relaxations and one post-relaxation on the fine level ($\omega = 0.7$), and two relaxations on the coarse level ($maxit =$ 4,000). The coarse-level aggregates were determined by the neighborhood aggregation method described in [5] using strength of connection in $A\,\mathrm{diag}(\mathbf{x}_i)$, with strength of connection parameter $\theta = 0.25$. We note that the aggregation transfer operator was frozen after the first iteration, and was used for all subsequent iterations.

The Bi-CGStab and GMRES algorithms were implemented according to the templates in [46]. Experiments with GMRES were run for subspaces of size $m =$ 10, 20, 25. We also considered the following preconditioners: ILU0, and ILUTP (ILU with thresholding and pivoting [6]) with drop tolerances $\{0.01, 0.001, 0.0001\}$. The preconditioners were constructed by Matlab's built-in sparse incomplete LU factorization method `ilu`. For each test problem we chose the Bi-CGStab and GMRES parameters that gave fastest execution time for the largest problem size ($maxit = 500$), and used that combination for all problem sizes. We note that in some instances the solution had very small nonpositive elements, in which case we took the absolute value and renormalized.

For the multilevel methods we performed at most 1,000 iterations, and used weighted Jacobi relaxation parameter $\omega = 0.7$, with at most $n_{coarse} = 12$ points on the coarsest level. For the OC-AGG methods (fixed and automatic) we used V(1,2) cycles ($\mu = 1$, $\nu_1 = 1$, $\nu_2 = 2$), and for MCAMG and AGG we used V(2,1) cycles. For tests involving the AGG method, the coarse degrees of freedom were determined by the Bottom-up aggregation technique described in [27], with aggregates frozen on all levels after the first cycle.

In the on-the-fly framework we used different parameters for the MCAMG and AGG algorithms, since the AGG setup is much cheaper than the MCAMG setup with respect to the cost of their corresponding additive cycles. For OTF-MCAMG we used convergence parameter $C = 1$ and threshold parameter $\varepsilon_\alpha = 10^{-4}$, with V(4,2) setup cycles and V(1,1) solution cycles. For all OTF-OC-AGG methods, we used convergence parameter $C = 0.7$ and threshold parameter $\varepsilon_\alpha = 10^{-5}$, with V(4,2) setup cycles and V(1,2) solution cycles. For the automatic over-correction, the relaxations in Equations (3.8) and (3.16) are not counted as post-relaxations, and therefore, these cycles are equivalent to V(1,3) cycles in terms of cost. These extra relaxations were performed with $\omega = 0.7$, unless stated otherwise. For OTF-AGG we used V(4,2) setup cycles, V(1,2) solution cycles, and $\varepsilon_\alpha = 10^{-4}$. We also used $C = 1$, since both the solution and setup cycles are less effective, and it is best to limit the number of setup cycles.

The various combinations of ν_1 and ν_2 described above were chosen to strike a balance between the convergence properties of the multilevel methods in question, and the work incurred by additional relaxation operations. In particular, for the automatic over-correction it was necessary to use $\nu_1 = 1$ and $\nu_2 = 2$ with the (OTF-)OC-AGG methods. This aids the minimization process (3.17) in choosing a better α since $\mathbf{x}_i$ is not significantly smoother than $\hat{\mathbf{x}}$ (otherwise, the minimizer might be $\alpha < 1$). Also, over-correction tends to introduce "noise" into the smooth approximation, which then needs to be smoothed away, so more post-relaxations are needed. As such, using a V(1,1) cycle with over-correction, for example, would be less appropriate. In the case of fixed over-correction, we note that there is no significant difference between applying V(1,2) or V(2,1) cycles. We mention that the initial guess was smoothed by 10 weighted Jacobi relaxations prior to iterating, which we counted as one multilevel iteration in the tables. Furthermore, when using the OTF framework, the iteration count included both setup cycles and solution cycles.

In the tables below we report the number of iterations to converge, *it*, the operator complexity, C_{op}, and the number of work units, WU. The operator complexity is defined as the sum of the number of nonzero elements in all operators, on all levels, divided by the number of nonzero elements in the fine-level operator A. This number gives a good indication of the amount of work required for a cycle and, for an optimal method, it should be bounded by a constant not too much larger than one as the problem size n increases. One work unit represents the amount of work needed to perform a single relaxation for the given problem size. As such, we can use work units as a measure of performance. Ideally, the number of work units should be small, and for a scalable method, we should observe a nearly constant number of work units as the problem size grows. The number of work units is obtained by dividing the total execution time by the time of one relaxation on the finest level. The main purpose of providing work units is to give insight into the scalability of each method as problem sizes increase. Work units also allow us to compare, for every test problem, the efficiency of the different methods as we have implemented them. However, absolute comparisons may not be as meaningful since they may depend significantly on the implementation details of each method. Additionally, the work-units measure is more accurate for larger problems since it is then less influenced by Matlab's compilation time and other system activities. We note that any blank entries in the tables below indicate that the method did not converge within the prescribed number of iterations. Furthermore, a dashed entry "–" indicates that the method diverged.

4.1. Tandem queue

The first test problem we consider is the tandem queueing network from [4], where two finite queues with single servers are placed in tandem. Customers arrive according to a Poisson distribution with rate μ, and the service time distribution at the two single-server stations is Poisson with rates μ_1 and μ_2. This is illustrated in Figure 1. The states of the system can be represented by tuples (n_1, n_2), where n_i is the number of customers waiting in the ith queue. We choose $(\mu,\ \mu_1,\ \mu_2) = (10, 11, 10)$ for the weights, which leads to a case of slow mixing. This is a periodic, nonsymmetric problem whose subdominant eigenvalue asymptotically approaches 1 at a rate inversely proportional to the problem size [3].

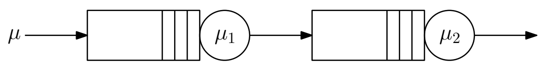

Figure 1. Tandem queueing network.

For GMRES and Bi-CGStab we used the ILUTP preconditioner with drop tolerances 0.0001 and 0.001, respectively. Their performance for the tandem queueing problem is given in Table I. In Table II we observe excellent results for the OTF-MCAMG method, with an essentially constant number of work units as the problem size grows. Furthermore, it is clear that OTF-MCAMG is competitive with Bi-CGStab and GMRES for sufficiently large problem sizes. Table III demonstrates the effectiveness of our over-correction approach. Comparing between AGG and OC-AGG we observe a large decrease in the number of iterations and works units, where OC-AGG is approximately 19 times faster than AGG for $n = 65536$. Similar behavior is witnessed for both fixed and automatic over-correction, in both OC-AGG and OTF-OC-AGG settings. We observe that the cost of OTF-OC-AGG is approximately 50% to 80% the cost of OC-AGG. Also, the (OTF-)OC-AGG methods performed slightly better with the fixed over-correction than with the automatic one.

Figure 2 shows a log-log plot of execution times for OTF-MCAMG, OTF-OC-AGG (automatic), Bi-CGStab and GMRES methods as a function of the problem size. We note that for Bi-CGStab and GMRES we have included the time required to construct the preconditioner, which is also reflected in the number of work units. For each method we observe a nearly linear relationship, where the slopes of the least-squares best fit lines are given by the bracketed numbers in the legend. OTF-MCAMG appears to scale better than all the methods presented, with a complexity that is nearly linear in the problem size. In the current implementation, it is clear that OTF-OC-AGG achieves better performance than the other methods presented, which is mainly attributable to its attractive operator complexity and simplicity. For example, each OTF-OC-AGG solution/setup cycle is much cheaper than a corresponding OTF-MCAMG solution/setup cycle.

Table I. Tandem queueing network. Results for weighted Jacobi, two-level aggregation, Bi-CGStab and GMRES(25). WU is the number of work units and *it* is the number of iterations to reduce the 1-norm of the initial residual by a factor of 10^8.

	one-level ($\omega = 0.99$)	two-level	Bi-CGStab		GMRES(25)	
n	*it*	*it*	*it*	WU	*it*	WU
4096	12005	3383	9	339	7	590
16384			15	629	11	1033
65536			31	889	17	1232
262144			62	1532	41	2051

Table II. Tandem queueing network. Results for multilevel aggregation (AGG), on-the-fly AGG, MCAMG and on-the-fly MCAMG. WU is the number of work units, C_{op} is the operator complexity, and *it* is the number of iterations to reduce the 1-norm of the initial residual by a factor of 10^8.

	AGG			OTF-AGG			MCAMG			OTF-MCAMG		
n	*it*	(C_{op})	WU	*it*	(C_{op})	WU	*it*	(C_{op})	WU	*it*	(C_{op})	WU
4096	159	(1.48)	5493	167	(1.48)	2691	11	(4.52)	2564	13	(4.53)	740
16384	268	(1.49)	8419	308	(1.49)	3068	13	(4.56)	3254	15	(4.57)	887
65536	456	(1.50)	12748	692	(1.50)	4771	13	(4.61)	2877	20	(4.65)	854
262144							14	(4.65)	2872	24	(4.67)	863

Table III. Tandem queueing network. Results for multilevel aggregation (AGG) and on-the-fly AGG each with automatic and fixed over-correction. WU is the number of work units, C_{op} is the operator complexity, and *it* is the number of iterations to reduce the 1-norm of the initial residual by a factor of 10^8.

	OC-AGG (automatic)			OC-AGG (fixed - 1.9)			OTF-OC-AGG (automatic)			OTF-OC-AGG (fixed - 1.9)		
n	*it*	(C_{op})	WU	*it*	(C_{op})	WU	*it*	(C_{op})	WU	*it*	(C_{op})	WU
4096	16	(1.48)	944	18	(1.48)	955	18	(1.48)	742	17	(1.48)	534
16384	18	(1.49)	744	19	(1.49)	868	18	(1.49)	601	18	(1.49)	487
65536	17	(1.50)	652	19	(1.50)	668	19	(1.50)	495	17	(1.50)	358
262144	18	(1.50)	682	18	(1.50)	614	23	(1.50)	564	17	(1.50)	339

4.2. MARCA ATM queue

The next test problem we consider is a multi-class, finite buffer, priority system. This model can be applied to telecommunications modeling, and has been used to model ATM queueing networks as discussed in [6]. For a complete description including all of the model parameters we refer to [4, 7]. We note that the model parameters were selected so that the resulting Markov chain is nearly completely decomposable, with the particular set of parameters given in [6]. The code and data files used to build the transition rate matrix corresponding to this Markov chain model are provided freely on the web [7].

Results for GMRES and Bi-CGStab were obtained using the ILUTP preconditioner with drop tolerances 0.001 and 0.01, respectively. We note that weighted Jacobi failed to converge within 20,000 iterations for any value of $\omega \in (0, 1]$. Similarly, the two-level aggregation method

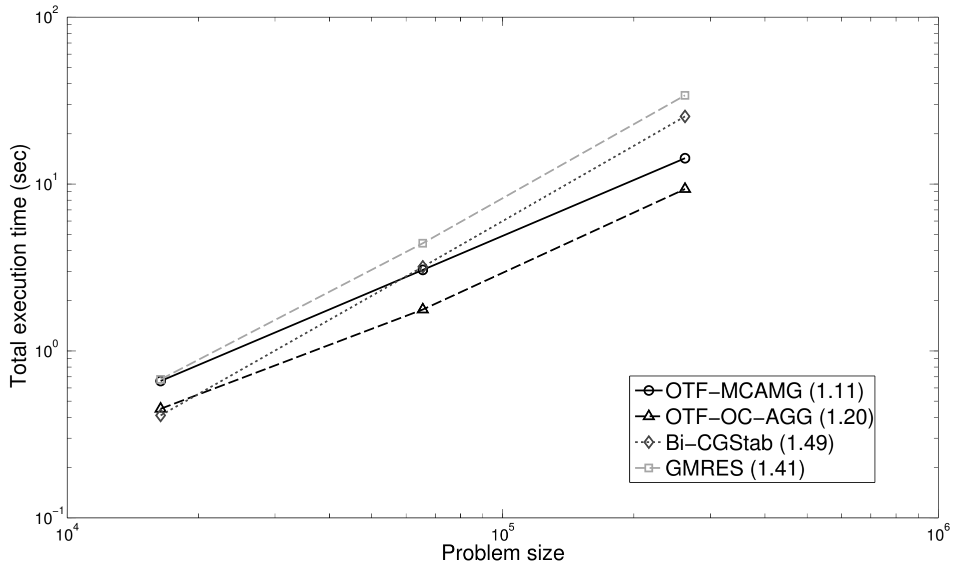

Figure 2. Tandem queueing network. Total execution times of preconditioned Bi-CGStab, preconditioned GMRES, OTF-MCAMG and OTF-OC-AGG (automatic) for $n = 16384$, 65536, 262144. The numbers in brackets are the slopes of the least-squares best fit lines, fitted through the data points.

also failed to converge within 4,000 iterations. As such they are not shown in Table IV below. In order to obtain satisfactory convergence for MCAMG and OTF-MCAMG it was necessary to use strength of connection parameter $\theta = 0.5$. Table IV gives the results for Bi-CGStab and

Table IV. MARCA ATM queueing network. Results for Bi-CGStab and GMRES(10). WU is the number of work units and *it* is the number of iterations to reduce the 1-norm of the initial residual by a factor of 10^8.

	Bi-CGStab		GMRES(10)	
n	*it*	WU	*it*	WU
1940	5	270	6	403
7956	8	165	7	201
32276	12	258	9	325
130068	22	322	14	386

GMRES, which perform quite well for this test problem. In Table V we observe that (OTF-)AGG performs surprisingly well for the MARCA ATM queue problem, and actually manages to compete with (OTF-)MCAMG in terms of work units. However, these results are somewhat misleading, since it was shown in [27] that the basic multilevel aggregation performs poorly for all other test cases tried, and really only performs well for the MARCA ATM queue problem. In general we expect (OTF-)MCAMG to be the superior method in terms of robustness and convergence, as demonstrated by the other test problems. The work units for OTF-AGG are approximately one third of those for AGG, and they appear to be constant as the problem

Table V. MARCA ATM queueing network. Results for multilevel aggregation (AGG), on-the-fly AGG, MCAMG and on-the-fly MCAMG. WU is the number of work units, C_{op} is the operator complexity, and *it* is the number of iterations to reduce the 1-norm of the initial residual by a factor of 10^8.

	AGG			OTF-AGG			MCAMG			OTF-MCAMG		
n	it	(C_{op})	WU	it	(C_{op})	WU	it	(C_{op})	WU	it	(C_{op})	WU
1940	34	(1.64)	1608	33	(1.64)	802	9	(4.69)	1783	8	(3.54)	751
7956	39	(1.55)	1379	51	(1.55)	613	10	(4.24)	1532	10	(3.86)	700
32276	61	(1.47)	1946	48	(1.47)	532	10	(4.43)	1876	12	(4.52)	840
130068	57	(1.44)	1836	58	(1.44)	571	10	(4.74)	1805	15	(5.32)	711

Table VI. MARCA ATM queueing network. Results for multilevel aggregation (AGG) and on-the-fly AGG each with automatic and fixed over-correction. WU is the number of work units, C_{op} is the operator complexity, and *it* is the number of iterations to reduce the 1-norm of the initial residual by a factor of 10^8.

	OC-AGG (automatic)			OC-AGG (fixed - 1.4)			OTF-OC-AGG (automatic)			OTF-OC-AGG (fixed - 1.4)		
n	it	(C_{op})	WU	it	(C_{op})	WU	it	(C_{op})	WU	it	(C_{op})	WU
1940	16	(1.64)	1035	17	(1.64)	1013	16	(1.64)	711	16	(1.64)	590
7956	19	(1.55)	907	22	(1.55)	906	19	(1.55)	472	22	(1.55)	417
32276	18	(1.47)	847	20	(1.47)	793	20	(1.47)	511	24	(1.47)	411
130068	20	(1.44)	893	22	(1.44)	864	22	(1.44)	562	24	(1.44)	427

size grows, which indicates mesh independent convergence (Table V).

In both the OC-AGG and OTF-OC-AGG settings, the fixed and automatic over-correction approaches again have similar iteration counts (Table VI). Clearly, the fixed over-correction has a slightly lower work unit count, since it does not calculate the parameter α on each level. For this test problem we observe a more moderate speedup of (OTF-)AGG using over-correction; both the fixed and automatic OC-AGG methods converge in about half the time it takes AGG to converge. As shown in Tables V and VI, the on-the-fly setting improves all multiplicative methods. However, whereas the cost of OC-AGG is half that of AGG, we observe that the cost of OTF-AGG is similar to the cost of OTF-OC-AGG in terms of their work units, while the latter requires about half the iterations to converge. This is mainly because both the fixed and automatic OTF-OC-AGG methods required more setup cycles than OTF-AGG.

Figure 3 shows the log-log plots of execution times for OTF-MCAMG, OTF-OC-AGG (automatic), Bi-CGStab and GMRES. While Bi-CGStab, GMRES and OTF-OC-AGG are faster than OTF-MCAMG, they do not scale as well as OTF-MCAMG. Thus, for sufficiently large problem sizes we expect that OTF-MCAMG will be superior to the other methods. In the current implementation, OTF-OC-AGG outperforms OTF-MCAMG in terms of solution time, however, it does not scale as well.

4.3. Directed random planar graph

The last test problem we consider is a random walk on an unstructured, directed planar graph. To construct the directed planar graph $\mathcal{D}$ we begin by randomly distributing n points in the unit square $(0,1) \times (0,1)$. These points are then connected via Delaunay triangulation, which

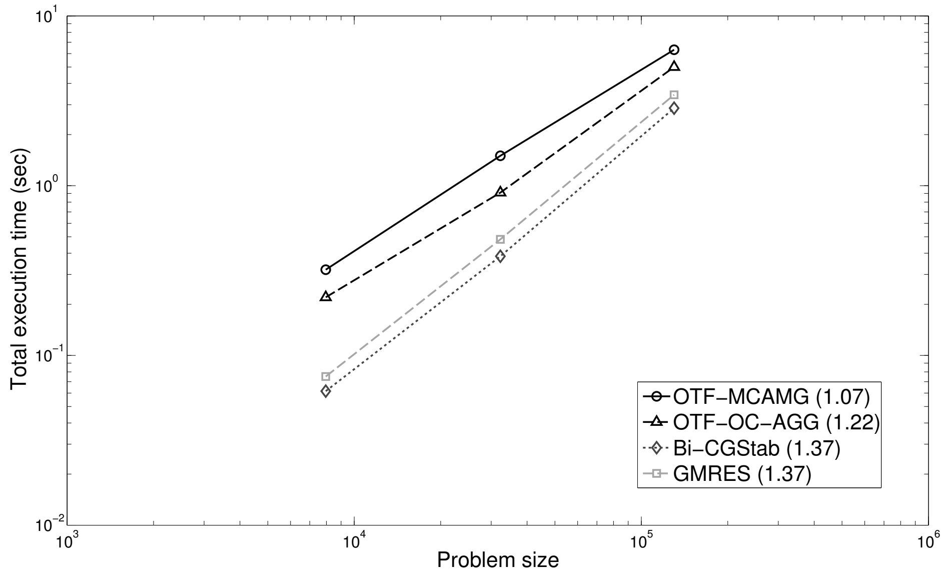

Figure 3. MARCA ATM queue. Total execution times of preconditioned Bi-CGStab, preconditioned GMRES, OTF-MCAMG and OTF-OC-AGG (automatic) for $n = 7956$, 32276, 130068. The numbers in brackets are the slopes of the least-squares best fit lines, fitted through the data points.

yields an undirected planar graph $\mathcal{G}$. To obtain a directed graph $\mathcal{D}$, we randomly select a set of edges from $\mathcal{G}$ and make them unidirectional. This is done in such a way that irreducibility is preserved (see [5] for further details regarding the construction of $\mathcal{D}$). A Markov chain is then obtained by performing a random walk on $\mathcal{D}$, where the probability of transitioning from node i to node j is given by the reciprocal of the number of outward arcs from node i. It is clear from the construction that the resulting Markov chain has a nonsymmetric sparsity structure. Furthermore, numerical computations of the transition matrices' spectra as the problem size grows confirms that this is a slowly mixing problem, and thus it is a prime candidate to test our multilevel approach.

Table VII. Directed random planar graph. Results for weighted Jacobi, two-level aggregation, Bi-CGStab and GMRES(25). WU is the number of work units and *it* is the number of iterations to reduce the 1-norm of the initial residual by a factor of 10^8.

	one-level ($\omega = 1$)	two-level	Bi-CGStab		GMRES(25)	
n	*it*	*it*	*it*	WU	*it*	WU
4096	9291	1625	8	307	12	305
16384			13	473	22	468
65536			23	560	45	679
262144			44	738	74	903

Results for Bi-CGStab and GMRES (see Table VII) were obtained using the ILUTP

Table VIII. Directed random planar graph. Results for multilevel aggregation (AGG), on-the-fly AGG, MCAMG and on-the-fly MCAMG. WU is the number of work units, C_{op} is the operator complexity, and *it* is the number of iterations to reduce the 1-norm of the initial residual by a factor of 10^8.

	AGG			OTF-AGG			MCAMG			OTF-MCAMG		
n	*it*	(C_{op})	WU	*it*	(C_{op})	WU	*it*	(C_{op})	WU	*it*	(C_{op})	WU
4096	151	(1.40)	5125	156	(1.40)	2137	12	(2.65)	1874	10	(2.65)	547
16384	305	(1.41)	8960	304	(1.41)	2418	13	(2.78)	2218	11	(2.78)	646
65536	594	(1.41)	16542	613	(1.41)	3695	14	(2.79)	2278	13	(2.79)	624
262144							15	(2.81)	2241	12	(2.81)	542

Table IX. Directed random planar graph. Results for multilevel aggregation (AGG) and on-the-fly AGG each with automatic and fixed over-correction. WU is the number of work units, C_{op} is the operator complexity, and *it* is the number of iterations to reduce the 1-norm of the initial residual by a factor of 10^8.

	OC-AGG (automatic)			OC-AGG (fixed - 1.7)			OTF-OC-AGG (automatic)			OTF-OC-AGG (fixed - 1.8)		
n	*it*	(C_{op})	WU	*it*	(C_{op})	WU	*it*	(C_{op})	WU	*it*	(C_{op})	WU
4096	18	(1.40)	846	20	(1.40)	893	17	(1.40)	468	20	(1.40)	409
16384	24	(1.41)	1026	24	(1.41)	904	20	(1.41)	345	23	(1.41)	333
65536	29	(1.41)	1090	30	(1.41)	1062	25	(1.41)	342	29	(1.41)	357
262144	40	(1.41)	1416	34	(1.41)	1150	29	(1.41)	384	33	(1.41)	351

preconditioner with drop tolerance 0.001. We note that it was necessary to use $\omega = 0.5$ for smoothing the CGC when determining α_{opt} for the automatic over-correction. Table IX shows substantial improvements using over-correction compared to basic multilevel aggregation, especially via the on-the-fly framework, where unlike in the previous cases, OTF-OC-AGG performed significantly better than OC-AGG. Again, the fixed and automatic over-correction approaches have comparable performance. Table VIII demonstrates good performance of the OTF-MCAMG method, which is further reflected in Figure 4, where nearly linear scaling is observed. As before, OTF-OC-AGG is faster due to its attractive operator complexity, while OTF-MCAMG is more scalable.

The over-correction results for all three test problems show that the automatic approach yields performance improvements that are generally as good as those obtained with a fixed α determined optimally via trial and error. This demonstrates the advantage of the automatic approach, since the α parameter does not have to be tuned. However, varying α on different levels as in the automatic approach does not seem to yield faster convergence compared to a fixed α on all levels and cycles. This deserves further investigation in the future to determine whether the automatic procedure can be improved. If not, then it may be sufficient to determine α automatically at the top level only, and use the same value for α at coarser levels.

5. Concluding remarks

In this paper we discussed a number of ways in which the basic multilevel aggregation algorithm for Markov chains of Horton and Leutenegger can be accelerated. As our main contribution

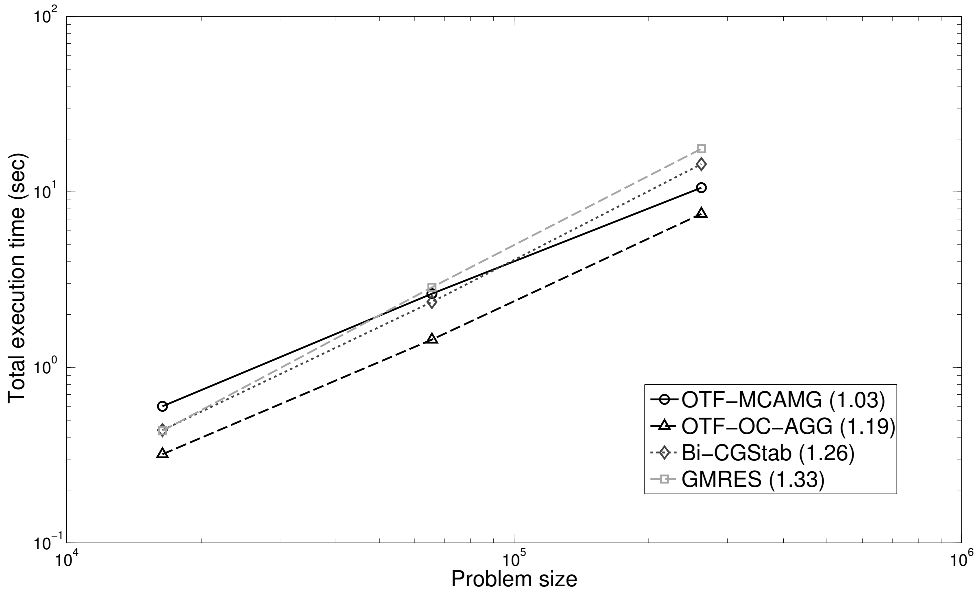

Figure 4. Directed random planar graph. Total execution times of preconditioned Bi-CGStab, preconditioned GMRES, OTF-MCAMG and OTF-OC-AGG (automatic) for n = 16384, 65536, 262144. The numbers in brackets are the slopes of the least-squares best fit lines, fitted through the data points.

we presented an automatic over-correction mechanism for both additive and multiplicative correction schemes for Markov chains. Numerical results demonstrated that the automatic over-correction approach cheaply and effectively improved the convergence of pure multilevel aggregation for Markov chains. Moreover, it was shown how further speedup was possible by using over-correction in conjunction with the on-the-fly adaptive framework. Indeed, the results showed that OC-AGG and OTF-OC-AGG are competitive with ILU-preconditioned Bi-CGStab and GMRES in terms of their work-unit measures. A similar observation was made for OTF-MCAMG, for which execution time scaling plots showed that it scaled better than the rest of the methods presented, with near-optimal performance in some cases. We mention that for the test problems considered, the conclusions regarding the relative merits of the methods in general hold for smaller convergence tolerances. Similar findings were obtained from numerical tests with convergence tolerances as small as $\tau = 10^{-12}$. While we did not consider other multilevel methods for Markov chains, including smoothed aggregation multigrid, recursively accelerated multilevel aggregation, and square and stretch multigrid, we expect that these methods would benefit similarly from the on-the-fly adaptive framework, and would be competitive with preconditioned Bi-CGStab and GMRES.

Acknowledgments

This work was supported in part by the Israeli Science Foundation, grant number 795/08, and by NSERC of Canada.

REFERENCES

1. Horton G, Leutenegger ST. A multi-level solution algorithm for steady-state Markov chains. *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 1994; 191–200.
2. Treister E, Yavneh I. On-the-fly adaptive smoothed aggregation multigrid for Markov chains. *SISC* 2010; doi:10.1137/100799034.
3. De Sterck H, Manteuffel TA, McCormick SF, Miller K, Ruge J, Sanders G. Algebraic multigrid for Markov chains. *SIAM J. Sci. Comput.* 2010; **32**:544–562.
4. Stewart WJ. *An Introduction to the Numerical Solution of Markov Chains*. Princeton University Press: Princeton, New Jersey, 1994.
5. De Sterck H, Manteuffel T, Miller K, Sanders G. Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains. *Adv. Comput. Math.* 2010; doi:10.1007/s10444-010-9168-x.
6. Philippe B, Saad Y, Stewart WJ. Numerical methods in Markov chain modeling. *Oper. Res.* 1992; **40**(6):1156–1179.
7. Stewart WJ. MARCA Models: A collection of markov chain models. Online. URL `http://www4.ncsu.edu/~billy/MARCA_Models/MARCA_Models.html`, last accessed August 2008.
8. Leutenegger ST, Horton G. On the utility of the multi-level algorithm for the solution of nearly completely decomposable Markov chains. *Technical Report 94-44*, ICASE 1994.
9. Krieger UR. Numerical solution of large finite Markov chains by algebraic multigrid techniques. *Computations with Markov Chains*, Stewart W (ed.), Kluwer: Boston, 1995; 403–424.
10. Takahashi Y. A lumping method for numerical calculations of stationary distributions of Markov chains. *Technical Report B-18*, Department of Information Sciences, Tokyo Institute of Technology 1975.
11. Simon HA, Ando A. Aggregation of variables in dynamic systems. *Econometrica* 1961; **29**:111–138.
12. Chatelin F, Miranker WL. Acceleration by aggregation of successive approximation methods. *Linear Algebra Appl.* 1982; **43**:17–47.
13. Mandel J, Sekerka B. A local convergence proof for the iterative aggregation method. *Linear Algebra Appl.* 1983; **51**:163–172.
14. Koury JR, McAllister DF, Stewart WJ. Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains. *SIAM J. Alg. Disc. Meth.* 1984; **5**(2):164–186.
15. Cao WL, Stewart WJ. Iterative aggregation/disaggregation techniques for nearly uncoupled Markov chains. *JACM* 1985; **32**(3):702–719.
16. Haviv M. Aggregation/disaggregation methods for computing the stationary distribution of Markov chains. *SIAM J. Numer. Anal.* 1987; **24**(4):952–966.
17. Krieger UR. On a two-level multigrid solution method for Markov chains. *Linear Algebra Appl.* 1995; **223–224**:415–438.
18. Marek I, Mayer P. Convergence analysis of an iterative aggregation/disaggregation method for computing stationary probability vectors of stochastic matrices. *Numer. Linear Algebra Appl.* 1998; **5**:253–274.
19. Marek I, Mayer P. Convergence theory of some classes of iterative aggregation/disaggregation methods for computing stationary probability vectors of stochastic matrices. *Linear Algebra Appl.* 2003; **363**:177–200.
20. Marek I, Szyld DB. Algebraic Schwarz methods for the numerical solution of Markov chains. *Linear Algebra Appl.* 2004; **386**:67–81.
21. Benzi M, Kuhlemann V. Restricted additive Schwarz methods for Markov chains. *Numer. Linear Algebra Appl.* 2000; **00**:1–20.
22. Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Second edn., SIAM: Philadelphia, Pennsylvania, 2003.
23. Buchholz P. Multilevel solutions for structured Markov chains. *SIAM J. Matrix Anal. Appl.* 2000; **22**(2):342–357.
24. De Sterck H, Manteuffel TA, McCormick SF, Nguyen Q, Ruge J. Multilevel adaptive aggregation for Markov chains, with application to web ranking. *SIAM J. Sci. Comput.* 2008; **30**:2235–2262.
25. De Sterck H, Manteuffel TA, McCormick SF, Miller K, Pearson J, Ruge J, Sanders G. Smoothed aggregation multigrid for Markov chains. *SIAM J. Sci. Comput.* 2010; **32**:40–61.

26. De Sterck H, Miller K, Sanders G, Winlaw M. Recursively accelerated multilevel aggregation for Markov chains. *SIAM J. Sci. Comput.* 2010; **32**(3):1652–1671.
27. Treister E, Yavneh I. Square and stretch multigrid for stochastic matrix eigenproblems. *Numer. Linear Algebr.* 2010; **17**:229–251.
28. Seibold B. Performance of algebraic multigrid methods for non-symmetric matrices arising in particle methods. *Numer. Linear Algebra Appl.* 2010; **17**:433–451.
29. Berman A, Plemmons RJ. *Nonnegative Matrices in the Mathematical Sciences*. SIAM: Philadelphia, Pennsylvania, 1987.
30. Briggs WL, Henson VE, McCormick SF. *A Multigrid Tutorial*. Second edn., SIAM: Philadelphia, Pennsylvania, 2000.
31. Brezina M, Manteuffel TA, McCormick SF, Ruge J, Sanders G. Towards adaptive smooth aggregation (αSA) for nonsymmetric problems. *SIAM J. Sci. Comput.* 2010; **32**:14–39.
32. Grassmann W, Taksar M, Heyman D. Regenerative analysis and steady-state distributions for Markov chains. *Oper. Res.* 1985; **33**(5):1107–1116.
33. Krieger UR, Müller-Clostermann B, Sczittnick M. Modeling and analysis of communication systems based on computational methods for Markov chains. *IEEE J. Selected Areas Commun.* 1990; **8**(9):1630–1648.
34. Brezina M, Falgout RD, MacLachlan S, Manteuffel TA, McCormick SF, Ruge J. Adaptive algebraic multigrid. *SIAM J. Sci. Comput.* 2006; **27**:1261–1286.
35. Brezina M, Falgout RD, MacLachlan S, Manteuffel TA, McCormick SF, Ruge J. Adaptive smoothed aggregation (αSA) multigrid. *SIAM J. Sci. Comput.* 2005; **25**:317–346.
36. Klaus Stüben. Algebraic multigrid (amg): an introduction with applications. *Technical Report 70*, GMD - Forschungszentrum Informationstechnik GmbH 1999.
37. Vaněk P, Míka S. Modification of two-level algorithm with overcorrection. *Appl. Math.* 1992; **37**:13–28.
38. Blaheta R. A multilevel method with overcorrection by aggregation for solving discrete elliptic problems. *J. Comput. Appl. Math.* 1988; **24**:227–239.
39. Braess D. Towards algebraic multigrid for elliptic problems of second order. *Computing* 1995; **55**:379–393.
40. Brandt A, Yavneh I. Accelerated multigrid convergence and high- Reynolds recirculating flows. *SIAM J. Sci. Comput.* 1993; **14**(3):607–626.
41. Guillard H, Vaněk P. An aggregation multigrid solver for convection-diffusion problems on unstructured meshes. *Technical Report UCD-CCM-130*, University of Colorado at Denver 1998.
42. Zhang J. Residual scaling techniques in multigrid, I: equivalence proof. *Appl. Math. Comput.* 1997; **86**:283–303.
43. Zhang J. Residual scaling techniques in multigrid, II: practical applications. *Appl. Math. Comput.* 1998; **90**:229–252.
44. Zhang J. Minimal residual smoothing in multi-level iterative method. *Appl. Math. Comput.* 1997; **84**:1–25.
45. Dayar T, Stewart WJ. Comparison of partitioning techniques for two-level iterative solvers on large, sparse, Markov chains. *SIAM J. Sci. Comput.* 2000; **21**:1691–1705.
46. Barrett R, Berry MW, Chan TF, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C, van der Vorst H. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM: Philadelphia, Pennsylvania, 1993.