Algebraic Multigrid for Markov Chains and Tensor Decomposition

by

Killian Miller

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, Canada, 2012

Abstract

The majority of this thesis is concerned with the development of efficient and robust numerical methods based on adaptive algebraic multigrid to compute the stationary distribution of Markov chains. It is shown that classical algebraic multigrid techniques can be applied in an exact interpolation scheme framework to compute the stationary distribution of irreducible, homogeneous Markov chains. A quantitative analysis shows that algebraically smooth multiplicative error is locally constant along strong connections in a scaled system operator, which suggests that classical algebraic multigrid coarsening and interpolation can be applied to the class of nonsymmetric irreducible singular M-matrices with zero column sums. Acceleration schemes based on fine-level iterant recombination, and over-correction of the coarse-grid correction are developed to improve the rate of convergence and scalability of simple adaptive aggregation multigrid methods for Markov chains. Numerical tests over a wide range of challenging nonsymmetric test problems demonstrate the effectiveness of the proposed multilevel method and the acceleration schemes.

This thesis also investigates the application of adaptive algebraic multigrid techniques for computing the canonical decomposition of higher-order tensors. The canonical decomposition is formulated as a least squares optimization problem, for which local minimizers are computed by solving the first-order optimality equations. The proposed multilevel method consists of two phases: an adaptive setup phase that uses a multiplicative correction scheme in conjunction with bootstrap algebraic multigrid interpolation to build the necessary operators on each level, and a solve phase that uses additive correction cycles based on the full approximation scheme to efficiently obtain an accurate solution. The alternating least squares method, which is a standard one-level iterative method for computing the canonical decomposition, is used as the relaxation scheme. Numerical tests show that for certain test problems arising from the discretization of high-dimensional partial differential equations on regular lattices the proposed multilevel method significantly outperforms the standard alternating least squares method when a high level of accuracy is required.

Acknowledgements

This thesis is the culmination of many years of hard work, and there are many people who deserve thanks for their encouragement and guidance. First and foremost, I would like to thank my supervisor Hans De Sterck. As my mentor Hans has exemplified the qualities that make a successful researcher, and has greatly influenced my development as a mathematician. I thank you for your inspiration, your dedication, and your patience, and for urging me to undertake a Ph.D. More than just a supervisor Hans has been a good friend as well as a strong competitor on the squash court.

I would also like to thank those people who influenced my early development as a mathematician. Thank you Mr. Carter and Mr. Bell, my high school mathematics teachers, for providing a solid foundation in the fundamentals of mathematics, and for sparking my interest in math. Special thanks to Dr. Rekha Srivastava at the University of Victoria, who opened my eyes to the beauty of mathematics and encouraged me to pursue a degree in mathematics. I would also like to thank Dr. Boualem Khouider at the University of Victoria for introducing me to applied mathematics research through summer research projects in climate modeling.

I owe a debt of gratitude to my colleagues, collaborators, and financial supporters. Thank you Geoffrey Sanders, Eran Treister, Irad Yavneh, Steve McCormick, Tom Manteuffel, and John Ruge, for your insightful ideas and your collaboration on a number of research projects. Thanks to my thesis committee members Steve Vavasis, Peter Forsyth, and Matt Scott for the time you have dedicated to my degree. Thank you to the University of Waterloo, the Department of Applied Mathematics, and the Natural Science and Engineering Research Council of Canada for their generous financial support. I would also especially like to recognize the Department of Applied Mathematics graduate secretary Helen Warren, who has worked tirelessly behind the scenes to keep our department running smoothly and to make our lives as graduate students a little easier.

Most importantly, I would like to thank my family. You have always believed in me, encouraged me, pushed me to follow my dreams, and have never failed to let me know that you are proud of me. There is no way in which I can fully express my gratitude for all that you have provided. Without your support I could not have achieved my goals. As much is this thesis is my work, it very much belongs to you as well. I would also like to thank my girlfriend Keeley for her patience and understanding during the writing of my thesis, and for her love and support.

Dedication

This thesis is dedicated to my parents, for their unwavering love and support.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The goal of this thesis is to develop efficient and robust numerical methods based on adaptive algebraic multigrid (AMG) for solving problems in linear and multilinear algebra. The research topics considered include the computation of the stationary distribution of large sparse Markov chains and the computation of the canonical decomposition of higher-order tensors. In what follows we introduce each of these topics separately, give a brief overview of this thesis, and conclude with a statement of research contributions.

1.1 Algebraic multigrid for Markov chains

A Markov chain is a stochastic process with a finite or countably infinite state space that satisfies a *memoryless* property, that is, the occurrence of a future state depends only on the present state and is independent of any past states. Markov chains come in two flavors: discrete-time chains in which state changes occur at discrete times and continuous-time chains in which state changes may occur at any point in time. In the case of a finite state space all the information of a Markov chain can be encoded into a matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$, referred to as a *transition matrix*. The ijth entry of this matrix gives the probability of transitioning to state j in the next time step given that the chain is currently in state i. The behavior of a Markov chain can be characterized by the various properties of its states, or in the case of a finite state space, by the various properties of its transition matrix. For example, a discrete-time Markov chain is called *irreducible* if every state can be reached from every other state, which is equivalent to its transition matrix being irreducible. Irreducible Markov chains with finite state spaces are of particular importance to this thesis because they are guaranteed to have a unique *stationary distribution*, that is, there exists a vector $\mathbf{x} \in \mathbb{R}^n$ with nonnegative components that sum to one such that $\mathbf{x} = \mathbf{P}\mathbf{x}$. In addition, if the states of the chain are aperiodic then the stationary distribution corresponds to the limiting probability distribution. This quantity is often of great interest in applications because it characterizes the behavior of the chain in the long run.

Markov chains are of interest in a wide range of applications including information retrieval and web ranking, performance modeling of computer and communication systems, dependability and security analysis, and analysis of biological systems [95, 100, 110]. For example, a well-known web ranking application is the PageRank algorithm used by the Google search engine. In its simplest interpretation the PageRank algorithm views the Internet as a large directed graph, where each node in the graph corresponds to a web page and each edge corresponds to a link between web pages. Essentially, the PageRank algorithm performs a random walk on this graph assigning edge weights that correspond to transition probabilities. The resulting edge-weighted graph corresponds to the transition matrix of a discrete-time Markov chain. Various regularizations are performed to manage any dangling nodes and to make the transition matrix irreducible and aperiodic. The page rank of each web page is then obtained by computing the stationary probability distribution of the regularized transition matrix. The exact details of how Google computes the stationary distribution is a closely guarded secret; however, the underlying method is allegedly a simple one-level stationary iterative method accelerated by vector extrapolation.

If the state space of a Markov chain is relatively small, direct methods such as LU factorization via Gaussian elimination followed by forward/backward substitution are effective for computing the stationary distribution (see [110] for an overview of direct methods applicable to Markov chains). However, in the case of Markov chains with large state spaces iterative methods are the only option for computing the stationary distribution. The simplest such method is the power method. One iteration of the power method consists of multiplying the current iterate by the transition matrix, followed by normalization. The rate of convergence of the power method can often be improved by considering other one-level solvers such as weighted Jacobi and successive overrelaxation [104], which may be viewed as preconditioned versions of the power method. While standard one-level methods are simple to implement and have provable convergence for certain classes of matrices, they can be unacceptably slow to converge when a *subdominant eigenvalue* of the transition matrix, that is, an eigenvalue of maximum magnitude less than one, is close to unity. We refer to a Markov chain as being *slowly mixing* if its subdominant eigenvalue with maximum real part approaches unity as the size of the state space grows (note that our definition may differ from others found in the literature). We are particularly interested in slowly mixing Markov chains because they represent a class of problems for which multigrid acceleration of standard one-level methods has the potential for large speedups.

A challenging class of problems in which some of the transition matrix eigenvalues tend to cluster around the eigenvalue $\lambda = 1$ are *nearly completely decomposable* (NCD) Markov chains. Nearly completely decomposable Markov chains are characterized by a state space that can be partitioned into disjoint subsets, with strong interactions among the states of a subset but weak interactions among the subsets themselves [91, 110]. Equivalently, the transition matrix can be symmetrically permuted to block form in which the nonzero elements of the off-diagonal blocks are small compared to those of the diagonal blocks. While one-level iterative methods typically perform poorly for NCD problems owing to a clustering of eigenvalues near $\lambda = 1$, two-level iterative aggregation/disaggregation (IAD) methods [34, 37, 39, 65, 80, 81, 82, 87, 88, 89, 114] are more effective for this class of Markov chains, oftentimes displaying rapid convergence to the exact solution. Iterative aggregation/disaggregation methods are multiplicative in nature and include geometric variants that select aggregates based on a priori knowledge of the Markov chain's structure and algebraic variants that select aggregates based on strength of connection in the problem matrix. As their name implies, IAD methods are closely related to aggregation-based multigrid methods in that the updated iterate after one iteration is obtained by disaggregating (interpolating) the approximate solution of a smaller (coarser) system of aggregated equations, followed by a few iterations of an inexpensive smoother. Iterative aggregation/disaggregation methods of the geometric type can obtain fast convergence for NCD problems by exploiting the a priori known block structure of the permuted transition matrix to guide the selection of the aggregated states. We mention that domain decomposition methods such as Schwarz methods [8, 90, 104], and preconditioned Krylov subspace methods including Arnoldi, GMRES, and biconjugate gradient [9, 10, 100, 104, 110] have also been successful at solving such problems. In fact, these methods constitute some of today's leading solvers for Markov chains.

Horton and Leutenegger were among the first to consider multilevel methods for computing the stationary distribution of Markov chains [69, 85]. Their method extends of the two-level iterative aggregation/disaggregation method for Markov chains developed by Takahashi [114], which makes use of the aggregated equations proposed by Simon and Ando [107]. Historically, multilevel methods with more than two levels have not been widely used for Markov chains, probably due to their poor convergence properties compared with two-level methods and the Krylov subspace methods mentioned above. However, with the recent advancements in multigrid over the past 20 years such as smoothed aggregation multigrid, bootstrap AMG, and the adaptive AMG framework, applying multigrid techniques to solve Markov chain problems shows much promise [15, 33, 99, 117, 118, 127]. Although theoretical convergence results are difficult to obtain for general nonsymmetric problems, empirical studies have demonstrated good convergence properties and robustness of multigrid methods applied to nonsymmetric linear systems [31, 38, 105, 112]. The goal of this thesis is to develop adaptive AMG methods for Markov chains that scale well and demonstrate good robustness. We note that while theoretical convergence results do exist for certain classes of two-level IAD methods [34, 81, 87, 88, 89], these results do not extend easily to multilevel methods. In fact, theoretical convergence results for AMG solvers applied to nonsymmetric problems are sparse in the literature, with advances having only recently been made [31, 96].

1.2 Algebraic multigrid for tensor decomposition

The second topic we consider is the canonical decomposition of higher-order tensors. An Nth-order tensor is an N-dimensional array of size $I_1 \times \cdots \times I_N$. The *order* of a tensor is the number of modes (dimensions). For example, a tensor of order three can be visualized as a rectangular prism of elements, while vectors and matrices are tensors of order one and two, respectively. The tensor canonical decomposition is a higher-order generalization of the matrix singular value decomposition (SVD) in that it decomposes a tensor as a sum of rank-one components. For example, if $\boldsymbol{\mathfrak{T}}$ is an Nth-order tensor of rank R, which implies that $\boldsymbol{\mathfrak{T}}$ can be expressed as a sum of no fewer than R rank-one components, then its canonical decomposition has the form

$$\boldsymbol{\mathfrak{T}} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \cdots \circ \mathbf{a}_r^{(N)},$$

where $\circ$ denotes the vector outer product. The rth rank-one component in the canonical decomposition is formed by taking the vector outer product of N column vectors $\mathbf{a}_r^{(n)} \in \mathbb{R}^{I_n}$ for $n = 1, \ldots, N$. For each mode n, the vectors $\mathbf{a}_1^{(n)}, \ldots, \mathbf{a}_R^{(n)}$ can be stored as the columns of an $I_n \times R$ matrix $\mathbf{A}^{(n)}$. This matrix is referred to as the mode-n *factor matrix*, and its columns are the mode-n *factors*. In this sense the factors are analogous to the singular vectors in the matrix SVD (with the singular values absorbed). However, a major difference is that the factors are not necessarily orthogonal in each mode.

We refer to the canonical decomposition as CANDECOMP/PARAFAC (CP) after the names originally given to it in early papers on the subject [35, 64]. In the example above the tensor rank was known, however, in general the rank is unknown. Computing the rank of a general tensor is an NP-hard problem [67], and so the aim is to find the "best" approximate decomposition (in some sense) for a given number of components R. The problem of computing the CP decomposition with R components that best approximates an arbitrary tensor $\mathfrak{Z}$ may be formulated as a nonlinear least squares optimization problem:

$$\text{minimize} \quad f(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}) \coloneqq \frac{1}{2}\Big\|\mathfrak{Z} - \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \cdots \circ \mathbf{a}_r^{(N)}\Big\|_F^2.$$

Here $\|\cdot\|_F$ denotes the Frobenius norm of a tensor, defined as the square root of the sum of the squared tensor elements. A general approach to solving this optimization problem is to find solutions of the *first-order optimality equations*, that is, to find a set of nontrivial factor matrices that zero out the gradient of f [1, 79].

The CP optimization problem defined above is nonconvex; consequently it may admit multiple local minima. Moreover, for any local minimizer there is a continuous manifold of equivalent minimizers [79]. This manifold arises because of a scaling indeterminacy inherent to the CP decomposition, that is, the individual factors composing each rank-one term can be rescaled without changing the rank-one term. The CP decomposition also exhibits a permutation indeterminacy in that the rank-one component tensors can be reordered arbitrarily [79]. The scaling and permutation indeterminacies can be removed by imposing a specific normalization and ordering of the factors. However, in spite of these steps the CP decomposition may still exhibit multiple local minima for some tensors, and depending on the initial guess, iterative methods for computing CP may converge to different stationary points. Furthermore, for certain tensors and certain values of R a best rank-R approximation does not exist [45]. For example, Kolda and Bader [79] give an example of a rank-three tensor that can be approximated arbitrarily closely by a tensor of rank two. Essentially, because the space of rank-two tensors is not closed one can build a sequence of rank-two tensors that converges to a tensor of rank other than two. Tensors that are approximated arbitrarily well by a decomposition of lower rank are called *degenerate*. Uniqueness of the exact CP decomposition up to scaling and permutation indeterminacies has been proved under mild conditions relating the ranks of the factor matrices with the tensor rank, and despite the aforementioned complications, CP is used in many different fields [79].

The primary application of CP is as a tool for data analysis, for example, as an alternative to the classical two-way principal component analysis, for which it has been used in a variety of fields including chemometrics, data mining, image compression, neuroscience, and telecommunications [79]. Alternatively, CP has been used to approximate tensors that arise from the discretization of integral and differential equations on high-dimensional regular lattices [75, 76, 97, 98], primarily as a form of *dimensionality reduction* to allow for more economical storage and efficient solution methods. Many algorithms have been proposed for computing the CP decomposition [1, 40, 43, 79, 101, 116]; however, the workhorse algorithm for computing the CP decomposition is still the original alternating least squares (ALS) method, first proposed in 1970 in early papers on CP [35, 64]. The alternating least squares method is simple to implement and often performs adequately; however, it can be slow to converge, and its convergence may depend strongly on the initial guess. Despite the simplicity and potential drawbacks of ALS, it has proved difficult over the years to develop alternative methods that significantly improve on ALS in a robust way for large classes of problems. Accordingly, ALS-type algorithms remain the method of choice in practice. Our goal is to accelerate the standard ALS algorithm by combining it with a nonlinear adaptive AMG framework in which ALS essentially acts as a relaxation method. We note that while the idea to apply multilevel methods to problems in multilinear algebra has already been discussed and analyzed [4, 16, 77], as far as we know the algorithm we propose is the first multigrid method for computing the canonical decomposition of a tensor.

1.3 Overview

Chapter 2 covers the pertinent background material that is necessary to understand the topics discussed in this thesis. We begin by defining the classes of nonnegative matrices and M-matrices, and provide sufficient conditions for irreducibility and aperiodicity of nonnegative matrices. In particular, we focus on the class of stochastic nonnegative matrices, which is fundamental to the study of Markov chains, as any transition probability matrix is necessarily a stochastic matrix. In §2.3 we provide a brief introduction to discrete-time and continuous-time Markov chains. Sufficient conditions that guarantee existence and uniqueness of the stationary and limiting distributions are given for homogeneous Markov chains with finite state spaces. Section 2.4 introduces tensors and the basic matrix and tensor operations that are needed when discussing the tensor canonical decomposition. Section 2.5, which is a precursor to our introduction of multigrid methods, discusses basic one-level iterative methods based on matrix splittings, and states conditions for their convergence. In particular, the application of these methods to irreducible singular M-matrices is investigated. We conclude Chapter 2 by introducing multigrid methods in §2.6. After discussing the fundamental underpinnings of all multigrid methods, we describe the classical AMG algorithm as well as aggregation-based AMG.

Chapters 3–6 contain the main contributions of this thesis. In Chapter 3 we describe an AMG method for computing the stationary distribution of an irreducible homogeneous Markov chain with finite state space. Our approach incorporates classical AMG coarsening and interpolation developed in the early stages of the AMG project by Brandt, McCormick, and Ruge [24] within a multiplicative correction scheme framework. We begin by deriving a two-level multiplicative correction scheme in an exact interpolation scheme [26] framework. In §3.3 we propose a modification of the classical AMG interpolation formula that produces a nonnegative interpolation operator with unit row sums. Sections 3.4 and 3.5 show how a *lumping* technique, which maintains the sign structure and irreducibility of the coarse-level system operators on all levels, results in strictly positive iterates on all levels and a fixed-point property for the exact solution. The connection between the multiplicative correction and additive correction frameworks is also discussed, which leads to a simple hybrid multiplicative/additive method. The chapter is concluded by numerical experiments for a variety of challenging test problems.

In Chapter 4 we discuss a simple method based on *iterant recombination* [119] to accelerate multigrid methods for computing the stationary distribution of Markov chains. Iterant recombination constructs an improved fine-level approximation as a linear combination of successive fine-level iterates from previous multigrid cycles, where the linear combination minimizes the residual with respect to some norm. Our acceleration method for Markov chains is different from standard applications of iterant recombination in that it must produce probability vectors. Consequently, after each multigrid cycle iterant recombination corresponds to solving a constrained minimization problem over a small subset of probability vectors. We consider both one-norm and two-norm minimization problems, and focus on accelerating the simple non-overlapping adaptive multilevel aggregation algorithm developed in [51]. We begin by the chapter describing the simple adaptive multilevel aggregation algorithm for Markov chains. In §4.2 we discuss the constrained iterant recombination approach, and in §4.3 we discuss Matlab's built-in quadratic programming solver `quadprog` as a solver for the two-norm iterant recombination optimization problem. We also describe an efficient algorithm for computing the analytical solution of the quadratic programming problem with window size two. Section 4.4 describes the ellipsoid method for nonlinear convex programs, and §4.5 discuss how it can be applied to solve the one-norm optimization problem that arises from the iterant recombination process. We also briefly discuss the connection between one-norm minimization and linear programming. The chapter is concluded by numerical experiments for a subset of the test problems considered in Chapter 3.

In Chapter 5 we discuss an approach to accelerate a simple non-overlapping multilevel aggregation for Markov chains that is based on scaling the coarse-grid correction by a scalar $\alpha > 1$. Borrowing the terminology of Míka and Vaněk [125] we refer to this technique as *over-correction*. In particular, we present an *automatic* over-correction mechanism, in which α is the solution of a simple minimization problem on each level. We compare automatic over-correction with a fixed over-correction approach in which a (nearly) optimal value of α is chosen a priori via trial and error. In addition, we compare multilevel aggregation accelerated by over-correction with unaccelerated multilevel aggregation, multilevel aggregation accelerated by iterant recombination from Chapter 4, the MCAMG method from Chapter 3, the preconditioned stabilized biconjugate gradient method [104], and the preconditioned generalized minimal residual method [104]. We begin Chapter 5 by briefly describing and motivating the classical over-correction mechanism for additive-correction multigrid applied to symmetric positive definite systems in §5.1. In particular, we also motivate the need for over-correction through a simple model problem. Section 5.2 describes our over-correction mechanism for multilevel aggregation applied to Markov chains. In §5.3 we present numerical results, and §5.4 contains concluding remarks.

In Chapter 6 we switch gears and discuss an adaptive AMG method for computing the canonical decomposition of higher-order tensors. The proposed method consists of two phases: an adaptive setup phase that uses a multiplicative correction scheme in conjunction with bootstrap algebraic multigrid (BAMG) interpolation [20, 72] to not only build the necessary transfer operators and coarse-level tensors, but also to compute initial approximations of the factor matrices, and a solve phase that uses additive correction cycles based on the full approximation scheme (FAS) [18, 23] to efficiently obtain an accurate solution. The method is adaptive in the sense that during the setup phase the transfer operators are continually improved by incorporating the most recent approximation of the desired factor matrices. We note that the combination of a multiplicative setup scheme and BAMG was previously considered in [83], where it formed the basis of an efficient eigensolver for multiclass spectral clustering problems. A similar approach was also proposed in [21, 72]. Furthermore, the multigrid framework we propose is closely related to recent work on an adaptive algebraic multigrid method for computing extremal singular triplets and eigenpairs of matrices [47], and to a lesser degree to multigrid methods for Markov chains [15, 53, 118]. We begin Chapter 6 by stating the first-order optimality equations for CP and describing the alternating least squares method. Section 6.2 describes the multilevel setup phase, and §6.3 describes the multilevel solve phase. Implementation details and numerical results are presented in §6.4, followed by concluding remarks.

Chapter 7 summarizes the work presented in this thesis and speculates on future avenues of research.

1.4 Statement of research contributions

The main contributions of this thesis are the development of adaptive multilevel methods for computing the stationary distribution of Markov chains and the canonical decomposition of higher-order tensors. Contributions to the literature for which I am a primary author are listed below along with the chapter of this thesis in which they are discussed. Names appearing in bold face type in the author lists indicate the primary authors.

Chapter 3:

- H. De Sterck, T. A. Manteuffel, S. F. McCormick, **K. Miller**, J. W. Ruge, and G. Sanders. Algebraic multigrid for Markov chains. *SIAM J. Sci. Comput.*, 32:544–562, 2010.

Chapter 4:

- H. De Sterck, T. Manteuffel, **K. Miller**, and **G. Sanders**. Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains. *Adv. Comput. Math.*, 35:375–403, 2010.
- H. De Sterck, **K. Miller**, and G. Sanders. Iterant recombination with one-norm minimization for multilevel Markov chain algorithms via the ellipsoid method. *Comput. Vis. Sci.*, 14:51–65, 2011.

Chapter 5:

- H. De Sterck, **K. Miller**, **E. Treister**, and I. Yavneh. Fast multilevel methods for Markov chains. *Numer. Linear Algebra Appl.*, 18:961–980, 2011.

Chapter 6:

- **H. De Sterck** and **K. Miller**. An adaptive algebraic multigrid algorithm for low-rank canonical tensor decomposition. *SIAM J. Sci. Comput.*, 35(1):B1–B24, 2013.

During my Ph.D. I also made smaller contributions to the following journal articles.

- H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Pearson, J. W. Ruge, and G. Sanders. Smoothed aggregation multigrid for Markov chains. *SIAM J. Sci. Comput.*, 32:40–61, 2010.
- H. De Sterck, K. Miller, G. Sanders, and M. Winlaw. Recursively accelerated multilevel aggregation for Markov chains. *SIAM J. Sci. Comput.*, 32(3):1652–1671, 2010.

Chapter 2

Background Material

2.1 Notation

Scalars are denoted by Roman letters and lowercase Greek letters, e.g., c, C, λ. Vectors are denoted by boldface lowercase letters, e.g., $\mathbf{v}$. An exceptional case is $\mathbf{1}$, which denotes the vector of all ones. Matrices are denoted by boldface capital letters, e.g., $\mathbf{A}$. The identity matrix is denoted by $\mathbf{I}$, and depending on the context, $\mathbf{0}$ denotes either the vector whose components are all zero or the matrix whose elements are all zero. Higher-order tensors (tensors of order three and higher) are denoted by boldface Euler script letters, e.g., $\boldsymbol{\mathscr{Z}}$. The ith component of a vector $\mathbf{x}$ is denoted by x_i, the ijth element of a matrix $\mathbf{A}$ is denoted by a_{ij}, and, for example, element (i, j, k, ℓ) of a fourth-order tensor $\boldsymbol{\mathscr{Z}}$ is denoted by $z_{ijk\ell}$. The jth column of a matrix $\mathbf{A}$ is denoted by $\mathbf{a}_j$. In the case of the identity matrix, $\mathbf{e}_j$ denotes its jth column, that is, the jth canonical basis vector. The kth element of a sequence is denoted by a superscript in parentheses, e.g., $\mathbf{x}^{(k)}$. In some cases we break this rule and use a subscript to denote the kth element of a sequence of scalars, e.g., a_k. In general, indices range from 1 to their capital versions, e.g., $k = 1, \dots, K$. Sets and spaces are denoted by capital Euler script letters, e.g., $\mathscr{T}$, except in a few cases. Given a square matrix $\mathbf{A}$, its *spectrum* is denoted by

$$\sigma(\mathbf{A}) = \{\text{set of eigenvalues of } \mathbf{A}\}, \tag{2.1}$$

and its *spectral radius* is denoted by

$$\rho(\mathbf{A}) = \max\{|\lambda| : \lambda \in \sigma(\mathbf{A})\}. \tag{2.2}$$

We use the operator $\operatorname{diag}(\cdot)$ in two different ways. If $\mathbf{v} \in \mathbb{R}^n$, then $\operatorname{diag}(\mathbf{v})$ is an $n \times n$ diagonal matrix with $v_1, \ldots, v_n$ on its diagonal. If $\mathbf{A} \in \mathbb{R}^{n\times n}$, then $\operatorname{diag}(\mathbf{A})$ is an $n \times n$ diagonal matrix with $a_{11}, \ldots, a_{nn}$ on its diagonal. We use $\operatorname{nnz}(\mathbf{A})$ to denote the number of nonzero elements in a sparse matrix $\mathbf{A}$. The standard inner product in $\mathbb{R}^n$ is denoted by $\langle \mathbf{x}, \mathbf{y}\rangle = \mathbf{x}^\top \mathbf{y}$. We make use of the usual vector norms: $\|\cdot\|_1$, $\|\cdot\|_2$, $\|\cdot\|_\infty$, and their corresponding induced matrix norms. As well, for any tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ the Frobenius norm is defined by

$$\|\boldsymbol{\mathcal{X}}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1,\ldots,i_N}^2}\,. \tag{2.3}$$

We note that if $\boldsymbol{\mathcal{X}}$ were replaced by a vector, then (2.3) would correspond to the vector two-norm, and if $\boldsymbol{\mathcal{X}}$ were replaced by a matrix, then (2.3) would correspond to the matrix Frobenius norm. Given a symmetric positive definite matrix $\mathbf{A}$, we also make use of the *energy inner product* and the *energy norm*, defined by

$$\langle \mathbf{u}, \mathbf{v}\rangle_{\mathbf{A}} = \langle \mathbf{A}\mathbf{u}, \mathbf{v}\rangle \quad \text{and} \quad \|\mathbf{u}\|_{\mathbf{A}} = \langle \mathbf{A}\mathbf{u}, \mathbf{u}\rangle^{1/2}.$$

2.2 Nonnegative matrices and M-matrices

We begin this section with a brief introduction to directed graphs. A directed graph or *digraph* is an ordered pair $\mathcal{D} = (\mathcal{N}, \mathcal{A})$ of sets $\mathcal{N}$ and $\mathcal{A}$, where $\mathcal{N} = \{v_1, \ldots, v_n\}$ is a nonempty set of nodes (vertices) and $\mathcal{A}$ is a set of ordered pairs of nodes called arcs. For nodes u and v in $\mathcal{N}$, a (directed) *u-v walk* in $\mathcal{D}$ is a finite set of nodes $u = v_0, v_1, \ldots, v_{k-1}, v_k = v$, beginning at u (start node) and ending at v (end node), such that $(v_{i-1}, v_i) \in \mathcal{A}$ for $i = 1, \ldots, k$. A (directed) *u-v path* in $\mathcal{D}$ is a directed u-v walk in which all nodes are distinct, except possibly the start and end nodes. The *length* of a path is defined as the number of arcs in the path. A (directed) *cycle* in $\mathcal{D}$ is a path in which the start node corresponds to the end node. A cycle of length one, which is just an arc that joins a node to itself is called a *loop*. For simplicity, when referring to a walk, a path or a cycle in a digraph, we omit the qualifier "directed". A digraph can be represented pictorially, where each node corresponds to a point and each arc corresponds to an arrowed line, indicating direction, between two points (see Figure 2.1). A digraph $\mathcal{D}$ is said to be *weakly connected*

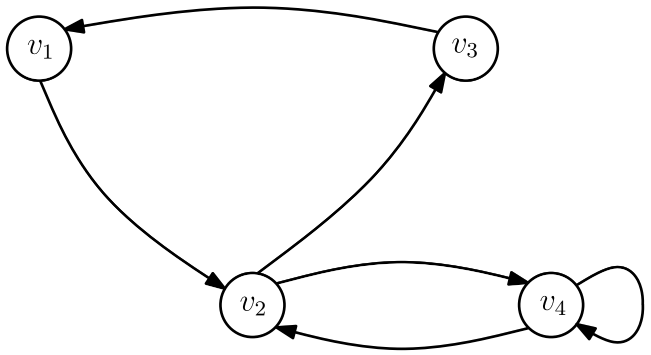


Figure 2.1: A simple digraph on four nodes. The sequence of node indices $(1, 2, 4, 2, 3)$ is an example of a walk, $(1, 2, 4)$ is an example of a path, and $(1, 2, 3, 1)$ is an example of a cycle. Note that node v_4 has a loop.

if there exists an undirected path (arcs can be traversed in either direction) between any two distinct nodes, whereas $\mathcal{D}$ is called *strongly connected* if there exists a directed path between any two distinct nodes. For example, the graph in Figure 2.1 is strongly connected. Another feature of a graph that we make use of is its *periodicity*. Formally, the periodicity of a strongly connected graph is defined as

$$p = \gcd\{\ell_1, \ell_2, \ldots, \ell_K\}, \tag{2.4}$$

where ℓ_k is the length of the kth cycle in $\mathcal{D}$ for $k = 1, \ldots, K$. If $p = 1$ the graph is said to be *aperiodic*, otherwise the graph is said to be *periodic* with period p. For example, the graph in Figure 2.1 has three cycles (ignoring permutations) of lengths 1, 2, and 3, hence it is aperiodic. Directed graphs can also be used to represent the nonzero structure of square matrices. For any matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ its associated graph is denoted by $\mathcal{D}(\mathbf{A}) = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \{1, \ldots, n\}$ and $(v_j, v_i) \in \mathcal{A}$ if $a_{ij} \neq 0$. For example, the graph in Figure 2.1 is the corresponding directed graph of the 4×4 matrix

$$\begin{bmatrix} & & * & \\ * & & & * \\ & * & & \\ & * & & * \end{bmatrix}$$

with nonzero elements indicated by asterisks. Many structural properties of a matrix such as periodicity and irreducibility can then be inferred from its corresponding graph.

We now discuss the various classes of matrices that are fundamental to this thesis. Because the transition probability matrix of a Markov chain necessarily has nonnegative elements, it is only appropriate that we begin by examining the class of nonnegative matrices. In what follows we provide conditions for a nonnegative matrix to be irreducible and aperiodic, and state the Perron–Frobenius theorem for irreducible nonnegative matrices. Using nonnegative matrices as our building blocks, we consider stochastic matrices and M-matrices. For an in-depth exploration of nonnegative matrices we recommend the books by Berman and Plemmons [11] and Horn and Johnson [68].

Definition 2.2.1 (Nonnegative matrix)**.** A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is said to be *nonnegative* if $a_{ij} \geq 0$ for all index pairs (i, j). Moreover, if $a_{ij} > 0$ for all index pairs (i, j), then $\mathbf{A}$ is called *positive.*

Similarly, we say that a vector $\mathbf{x}$ is nonnegative if $x_i \geq 0$ for all i, and is positive (or strictly positive) if $x_i > 0$ for all i. Also, we say that a vector $\mathbf{x}$ or matrix $\mathbf{A}$ is *nonpositive* if $-\mathbf{x}$ or $-\mathbf{A}$ is nonnegative. We now prove a property of nonnegative matrices that leads to a lower bound on their spectral radius.

Lemma 2.2.1. *Let $\|\cdot\|$ be any matrix norm on $\mathbb{C}^{n\times n}$. Then*

$$\rho(\mathbf{A}) = \lim_{k\to\infty} \|\mathbf{A}^k\|^{1/k}$$

for any matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$.

Proof. See Corollary 5.6.14 in [68]. $\square$

Theorem 2.2.1. *Let $\mathbf{A}$ and $\mathbf{B}$ be nonnegative matrices in $\mathbb{R}^{n\times n}$ such that $a_{ij} \le b_{ij}$ for all index pairs (i, j). Then $\rho(\mathbf{A}) \le \rho(\mathbf{B})$.*

Proof. For some integer $k \ge 1$ suppose that $\mathbf{A}^k \le \mathbf{B}^k$ where the inequality is applied elementwise. Then for any index pair (i, j) we have

$$(\mathbf{A}^{k+1})_{ij} = \sum_{\ell=1}^{n} a_{i\ell}(\mathbf{A}^k)_{\ell j} \le \sum_{\ell=1}^{n} b_{i\ell}(\mathbf{B}^k)_{\ell j} = (\mathbf{B}^{k+1})_{ij}.$$

Therefore, by induction, $\mathbf{A}^k \le \mathbf{B}^k$ for all $k = 1, 2, \ldots$, and hence

$$\|\mathbf{A}^k\|^{1/k} \le \|\mathbf{B}^k\|^{1/k}$$

for all natural numbers k. Taking the limit as $k \to \infty$ gives $\rho(\mathbf{A}) \le \rho(\mathbf{B})$ by Lemma 2.2.1. $\square$

For any nonnegative matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, Theorem 2.2.1 implies that

$$\max_{i=1,\ldots,n} a_{ii} = \rho(\operatorname{diag}(\mathbf{A})) \le \rho(\mathbf{A}).$$

Irreducibility is a structural property of a matrix in that it depends only on the location of the nonzero matrix elements and not on their values. Equivalent conditions for a nonnegative matrix to be irreducible are given by Theorem 2.2.2; see [68] for more details.

Theorem 2.2.2 (Irreducibility)**.** *A nonnegative matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is said to be irreducible if any of the following equivalent conditions are satisfied.*

1. *The digraph $\mathcal{D}(\mathbf{A})$ is strongly connected.*
2. *For each index pair (i, j) there exists a natural number m such that $(\mathbf{A}^m)_{ij} > 0$.*
3. *Each element of $(\mathbf{I} + \mathbf{A})^{n-1}$ is strictly positive.*
4. *The transpose $\mathbf{A}^\top$ is irreducible.*

We note that by the above theorem any matrix whose elements are all nonzero is necessarily irreducible. A matrix that is not irreducible is said to be *reducible* and is characterized by its graph failing to be strongly connected. Equivalently, a reducible matrix can be put into block lower triangular form by a symmetric permutation of its rows and columns. We do not concern ourselves with reducible matrices in this thesis; for more information see [11]. See also [110] for a discussion of reducible matrices as they pertain to Markov chains.

The set of *primitive matrices* is equivalent to the set of irreducible aperiodic nonnegative matrices. The following theorem (see [68]) states sufficient conditions for an irreducible nonnegative matrix to be aperiodic, and hence primitive.

Theorem 2.2.3 (Primitive matrix)**.** *An irreducible nonnegative matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is said to be primitive if any of the following conditions are satisfied.*

1. *The greatest common divisor of the lengths of any directed cycle in $\mathcal{D}(\mathbf{A})$ is unity.*
2. *There exists a natural number m such that $\mathbf{A}^m$ has strictly positive elements.*
3. *The matrix $\mathbf{A}$ has only one eigenvalue of maximum modulus.*
4. *The trace of $\mathbf{A}$ is positive, i.e., $a_{ii} > 0$ for some index i.*

The most significant result concerning irreducible nonnegative matrices is the Perron–Frobenius theorem which characterizes the eigenspace of this class of matrices. Details of its proof may be found in [11, 68, 126]. We note that versions of the Perron–Frobenius theorem also exist for positive matrices and primitive matrices [68].

Theorem 2.2.4 (Perron–Frobenius)**.** *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be an irreducible nonnegative matrix.*

1. *There exists a positive real eigenvalue of $\mathbf{A}$ that is equal to the spectral radius $\rho(\mathbf{A})$.*
2. *To $\rho(\mathbf{A})$ there corresponds a right eigenvector $\mathbf{x}$, unique up to scaling, which can be chosen such that all components of $\mathbf{x}$ are positive.*
3. *The spectral radius $\rho(\mathbf{A})$ is an algebraically simple eigenvalue of $\mathbf{A}$.*
4. *If $\mathbf{A}$ has exactly p eigenvalues equal in modulus to $\lambda_1 = \rho(\mathbf{A})$, then these eigenvalues are distinct and are the roots of the equation $\lambda^p - \lambda_1{}^p = 0$, that is,*

$$\lambda = \lambda_1 \exp(2\pi i k/p) \quad \textit{for} \quad k = 0, \dots, p-1.$$

An important subclass of the nonnegative matrices that pertains to the study of Markov chains are the stochastic matrices. In particular, the transition matrices of finite state Markov chains, which encode all the information of the chain, are stochastic matrices. A nonnegative matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is said to be *column-stochastic* if each of its columns sum to unity. In this thesis we assume that all transition matrices are column-oriented, and hence column-stochastic. Some useful properties of stochastic matrices are presented below in the form of a theorem.

Theorem 2.2.5 (Properties of stochastic matrices)**.** *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a stochastic matrix.*

1. *The left eigenvector of $\mathbf{A}$ corresponding to a unit eigenvalue is $\mathbf{1}$ (because the columns of $\mathbf{A}$ sum to one).*
2. *The eigenvalue $\lambda_1 = 1$ is a dominant eigenvalue of $\mathbf{A}$, that is, $\rho(\mathbf{A}) = \lambda_1$.*
3. *If $\mathbf{A}$ is irreducible then $\lambda_1 = 1$ is a simple eigenvalue whose corresponding eigenvector has strictly positive elements that sum to one.*

Proof. Property 1 follows by the fact that $\mathbf{A}$ has unit column sums. Property 2 follows by the fact that $\mathbf{1}^\top \mathbf{P} = \mathbf{1}^\top$ and $\rho(\mathbf{A}) \leq \|\mathbf{A}\|_1 = 1$. Property 3 is a direct consequence of the Perron–Frobenius theorem. □

M-matrices have a multitude of definitions, in fact, Berman and Plemmons list fifty equivalent definitions for nonsingular M-matrices (see Chapter 6, Theorem 2.3 in [11]). Consequently, we state the definition which is perhaps the most natural and certainly the most useful in this thesis.

Definition 2.2.2 (M-matrix)**.** The matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is an M-matrix if there exists an irreducible nonnegative matrix $\mathbf{B} \in \mathbb{R}^{n\times n}$ such that $\mathbf{A} = \gamma\mathbf{I} - \mathbf{B}$ for some scalar $\gamma \geq \rho(\mathbf{B})$.

If the scalar γ in Definition 2.2.2 is strictly greater than the spectral radius of $\mathbf{B}$ then $\mathbf{A}$ is a *nonsingular M-matrix*, otherwise, if $\gamma = \rho(\mathbf{A})$ then $\mathbf{A}$ is a *singular M-matrix*. M-matrices have nonpositive off-diagonal elements, and because the maximum diagonal element of a nonnegative matrix is less than or equal to its spectral radius, they have nonnegative diagonal elements. In particular, nonsingular M-matrices have strictly positive diagonal elements. M-matrices arise naturally in the study of Markov chains, for example, see Proposition 2.3.1 in §2.3. Below we present some properties of singular and nonsingular M-matrices (consult [11] and [126] for proofs).

Theorem 2.2.6 (Properties of singular M-matrices)**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a singular M-matrix.*

1. *If $\mathbf{A}$ is irreducible then* $\text{null}(\mathbf{A}) = \text{span}(\mathbf{x}_r)$ *and* $\text{null}(\mathbf{A}^\top) = \text{span}(\mathbf{x}_\ell)$*, where the vectors $\mathbf{x}_r$ and $\mathbf{x}_\ell$ are strictly positive.*

2. *If $\mathbf{B} \in \mathbb{R}^{n\times n}$ has a strictly positive vector in its left or right null space, and if its off-diagonal elements are nonpositive, then $\mathbf{B}$ is a singular M-matrix.*

3. *If $\mathbf{A}$ is irreducible then every principal submatrix of $\mathbf{A}$ other than itself, is a nonsingular M-matrix.*

4. *If $\mathbf{A}$ is irreducible then it has strictly positive diagonal elements.*

5. *The real part of each nonzero eigenvalue of $\mathbf{A}$ is positive.*

Theorem 2.2.7 (Properties of nonsingular M-matrices)**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a nonsingular M-matrix.*

1. *The diagonal elements of $\mathbf{A}$ are strictly positive.*
2. *The inverse $\mathbf{A}^{-1}$ is nonnegative.*
3. *If $\tilde{\mathbf{A}}$ is obtained from $\mathbf{A}$ by setting any of its off-diagonal elements to zero, then $\tilde{\mathbf{A}}$ is also an M-matrix.*
4. *If $\mathbf{A}$ is symmetric then it is positive definite. Moreover, if $\mathbf{A}$ is a symmetric positive definite matrix with nonpositive off-diagonal elements, then it is a nonsingular M-matrix.*

2.3 Markov chains

This section presents a brief introduction to discrete-time and continuous-time Markov chains. In particular, sufficient conditions that guarantee existence and uniqueness of the stationary distribution are given for Markov chains with time-independent transitions and finite state spaces. A terse introduction to Markov chain theory is given by Norris [95], and a comprehensive introduction to numerical methods for Markov chains is given by Stewart [110].

A *stochastic process* is defined as a family of random variables $\{X(t) : t \in \mathcal{T}\}$ defined on a given probability space and indexed by a time parameter t, where t varies over some index set (parameter space) $\mathcal{T}$. The values assumed by the random variable $X(t)$ are called *states*, and the set of all possible states, denoted by $\mathcal{S}$, is the *state space* of the process. The state space and the parameter space may be either discrete or continuous. For example, the stochastic process corresponding to the temperature outside at specific times during the day has a discrete parameter space and a continuous state space. Processes for which the parameter space is discrete (continuous) are referred to as discrete-time (continuous-time) processes.

A *Markov process* is a special type of stochastic process characterized by a conditional probability density that satisfies a *memoryless* property (Markov property), that is, the occurrence of a future state depends only upon the present state and is independent of any past states. More precisely, $X(t)$ is a continuous-time Markov process if for all integers n, and any sequence $t_0, t_1, \ldots, t_n$ ordered so that $t_0 < t_1 < \cdots < t_n < t$, we have

$$\mathbb{P}(X(t) \leq x \,|\, X(t_n) = x_n, \ldots, X(t_0) = x_0) = \mathbb{P}(X(t) \leq x \,|\, X(t_n) = x_n). \tag{2.5}$$

It is evident from (2.5) that $X(t_n)$ contains all the relevant information concerning the history of the process. We note that if transitions from a state $X(t)$ depend on the time t then the Markov process is said to be *nonhomogeneous*. Conversely, if transitions do not depend on the time parameter then the Markov process is said to be *homogeneous*. A Markov process whose state space is discrete (at most countably infinite) is referred to as a *Markov chain*, where, without loss of generality, the state space is represented by a subset of the positive integers. The numerical methods discussed in this thesis are intended for homogeneous Markov chains with finite state spaces $\mathcal{S} = \{1, 2, \ldots, N\}$. Therefore, the definitions and theoretical results in this section are presented for this class of Markov chains. We note that Markov chains can be conveniently described by a *transition diagram*, which is a weighted directed graph in which the nodes of the graph correspond to the states, arcs correspond to transitions between states, and weights correspond to transition rates or transition probabilities (see Figure 2.2).

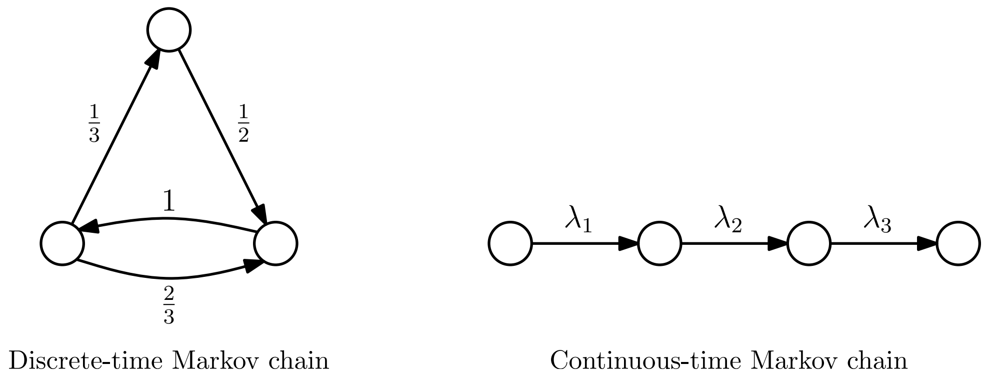


Figure 2.2: Illustration of transition diagrams for some arbitrary Markov chains.

Without loss of generality, the parameter space of a discrete-time Markov chain (DTMC) can be represented by the set of natural numbers $\mathcal{T} = \mathbb{N} = \{0, 1, 2, \ldots\}$, where the nth observation defines the random variable X_n. A DTMC satisfies the following Markov property for all $n \in \mathcal{T}$ and all states x_n:

$$\mathbb{P}(X_{n+1} = x_{n+1} \,|\, X_n = x_n, \ldots, X_0 = x_0) = \mathbb{P}(X_{n+1} = x_{n+1} \,|\, X_n = x_n). \tag{2.6}$$

The (single-step) *transition probabilities* of a DTMC, denoted by

$$p_{ij} = \mathbb{P}(X_{n+1} = i \,|\, X_n = j) \quad \text{for all} \quad n = 0, 1, 2, \ldots,$$

are the conditional probabilities of making a transition from state j to state i as the time parameter increases from n to $n+1$. Arranging the transition probabilities into an array, we arrive at the $N \times N$ *transition probability matrix* for the chain defined by $(\mathbf{P})_{ij} = p_{ij}$. Since

$$0 \le p_{ij} \le 1 \quad \text{for} \quad i, j = 1, \ldots, N \qquad \text{and} \qquad \sum_{i=1}^{N} p_{ij} = 1 \quad \text{for} \quad j = 1, \ldots, N,$$

it follows that $\mathbf{P}$ is a column-stochastic matrix (see §2.2). The single-step transition probabilities can be generalized by considering the conditional probability of transitioning to state i in $n \ge 1$ time steps given that the chain is currently in state j. What arises are the n-step transition probabilities, denoted by

$$p_{ij}^{(n)} = \mathbb{P}(X_{m+n} = i \,|\, X_m = j) \quad \text{for all} \quad m = 0, 1, 2, \ldots.$$

Analogously, the n-step transition probability matrix is defined by $(\mathbf{P}^{(n)})_{ij} = p_{ij}^{(n)}$. An important property satisfied by the n-step transition probability matrix is given by the *Chapman–Kolmogorov* equations:

$$\mathbf{P}^{(n)} = \mathbf{P}^{(n-k)}\mathbf{P}^{(k)} \quad \text{for} \quad 0 < k < n. \tag{2.7}$$

This result states that n-step transition probabilities can be written as the sum of products of k-step and $(n-k)$-step transition probabilities. In particular, (2.7) can be used to show that $\mathbf{P}^{(n)} = \mathbf{P}^n$, that is, the n-step transition probability matrix is obtained by multiplying the single-step transition probability matrix by itself n times. Moreover, $\mathbf{P}^{(n)}$ is a column-stochastic matrix.

We next consider probability distributions defined on the states of a discrete-time Markov chain. We denote by $u_i^{(n)}$ the probability that the Markov chain is in state i at time step n, where it is clear that for any $n \geq 0$

$$u_i^{(n)} \in \mathbb{R}, \qquad 0 \leq u_i^{(n)} \leq 1 \;\text{ for }\; i = 1, \ldots, N, \quad \text{and} \quad \sum_{i=1}^{N} u_i^{(n)} = 1.$$

In vector format we obtain the column-vector defined by $(\mathbf{u}^{(n)})_i = u_i^{(n)}$. The distribution of the chain at time step $n+1$ is obtained by multiplying the current distribution at time step n by the transition probability matrix, i.e., $\mathbf{u}^{(n+1)} = \mathbf{P}\mathbf{u}^{(n)}$. It follows that if $\mathbf{u}^{(0)}$ is the initial distribution of the chain then

$$\mathbf{u}^{(n)} = \mathbf{P}^n \mathbf{u}^{(0)} = \mathbf{P}^{(n)} \mathbf{u}^{(0)} \;\text{ for all }\; n = 1, 2, 3, \ldots.$$

The *stationary distribution* of a DTMC is a probability distribution $\mathbf{u}$ such that $\mathbf{u} = \mathbf{P}\mathbf{u}$. The stationary distribution of a finite DTMC exists and is unique if the chain is irreducible.

Definition 2.3.1 (Irreducibility)**.** A finite DTMC is irreducible if its transition matrix is irreducible.

Because any stochastic matrix is necessarily nonnegative with spectral radius equal to unity (see §2.2), existence and uniqueness of the stationary distribution follows by Perron–Frobenius theorem for irreducible nonnegative matrices; see Theorem 2.2.4. Moreover, the Perron–Frobenius theorem implies that the components of the stationary distribution are strictly positive. We note that if the transition matrix is not irreducible there may exist multiple eigenvectors associated with the unit eigenvalue each of which can have zero or negative components [110]. Given some initial probability distribution $\mathbf{u}^{(0)}$, the limit

$$\mathbf{u} = \lim_{n\to\infty} \mathbf{u}^{(n)} \tag{2.8}$$

if it exists is referred to as the *limiting distribution*. The limiting distribution of a finite irreducible DTMC exists and is independent of the initial distribution if the chain is aperiodic. Moreover, the limiting distribution, if it exists, is identical to the stationary distribution.

Definition 2.3.2 (Periodicity)**.** A finite irreducible DTMC is periodic with period p if the directed graph $\mathcal{D}(\mathbf{P})$ corresponding to its transition matrix $\mathbf{P}$ has period p. If $p = 1$ the Markov chain is aperiodic.

To see why aperiodicity is essential, let X be an irreducible Markov chain with period $d > 1$. Because X is periodic with period d it follows that $p_{jj}^{(n)} = 0$ for any state j whenever n is not a multiple of d. Now consider the sequence of points given by $n_k = dk + 1$ for $k \geq 1$. Since n_k is not a multiple of d for any k, the limit

$$\lim_{k\to\infty} \mathbb{P}(X_{n_k} = j \text{ and } X_0 = j) = \lim_{k\to\infty} p_{jj}^{(n_k)}\, \mathbb{P}(X_0 = j) = 0$$

for any state j. Therefore, the limit of $\mathbb{P}(X_n = j \text{ and } X_0 = j)$ as $n \to \infty$ does not approach some $u_j > 0$ along the subsequence $\{n_k\}_{k\geq 1}$. Consequently, there cannot exist a strictly positive limiting distribution that is independent of the initial distribution.

We now turn our attention to continuous-time Markov chains (CTMC), which arise when a state change can occur at any point in time. To simplify matters we consider homogeneous CTMCs with finite state spaces. A stochastic process $\{X(t) : t \geq 0\}$ is a continuous-time Markov chain if for all integers $n \geq 0$, and for any sequence $t_0, t_1, \ldots, t_n, t_{n+1}$ such that $t_0 < t_1 < \cdots < t_n < t_{n+1}$, we have

$$\mathbb{P}(X(t_{n+1}) = x_{n+1} \,|\, X(t_n) = x_n, \ldots, X(t_0) = x_0) = \mathbb{P}(X(t_{n+1}) = x_{n+1} \,|\, X(t_n) = x_n). \tag{2.9}$$

Analogous to discrete-time Markov chains the transition probabilities for any time t are defined by

$$p_{ij}(\tau) = \mathbb{P}(X(t+\tau) = i \,|\, X(t) = j),$$

where

$$\sum_{i=1}^{N} p_{ij}(\tau) = 1 \quad \text{for all} \quad \tau \geq 0.$$

The transition probability $p_{ij}(\tau)$ is the conditional probability of being in state i after an interval of length τ, given that the chain is currently in state j. Whereas a discrete-time Markov chain is represented by a matrix of transition probabilities, a continuous-time Markov chain is represented by a matrix of transition rates $\mathbf{Q}$, referred to as the *transition rate matrix* or the *infinitesimal generator*. The ijth element of $\mathbf{Q}$ is the (instantaneous) rate at which transitions occur from state j to state i at any time t. Under the assumption of a finite state space, and as $\tau \to 0$ uniformly in t, for all i

$$\begin{aligned} p_{ij}(\tau) &= q_{ij}\tau + o(\tau) \quad \text{for} \quad i \neq j, \\ p_{ii}(\tau) &= 1 + q_{ii}\tau + o(\tau), \end{aligned}$$

where the Landau notation $o(\tau)$ means that $o(\tau)/\tau \to 0$ as $\tau \to 0$. For $i \neq j$ this expression says that correct to terms of order $o(\tau)$, the probability that a transition occurs from state j to state i within τ time units is equal to the rate of transition multiplied by the length of the time interval. By conservation of total probability, for each i

$$p_{ii}(\tau) = 1 - \sum_{i \neq j} p_{ij}(\tau) \quad \Rightarrow \quad q_{ii} = -\sum_{i \neq j} q_{ij} + \frac{o(\tau)}{\tau} \quad \text{for all} \quad \tau > 0. \tag{2.10}$$

Therefore, taking the limit as $\tau \to 0^+$, the diagonal entries of $\mathbf{Q}$ are given by

$$q_{ii} = -\sum_{i \neq j} q_{ij} \quad \text{for} \quad i = 1, \ldots, N. \tag{2.11}$$

Intuitively, because the probability of transitioning to a different state increases as the time interval grows larger (i.e., $q_{ij} \geq 0$), the probability of remaining at that state must decrease with time, and therefore the corresponding transition rate should be negative. Consequently, the transition rate matrix has negative diagonal elements and nonnegative off-diagonal elements. Moreover, $\mathbf{Q}$ is *weakly diagonally dominant* because

$$\sum_{i=1}^{N} |q_{ij}| = |q_{jj}| \quad \text{for} \quad j = 1, \ldots, N.$$

Many properties of a continuous-time Markov chain can be deduced from its corresponding *embedded Markov chain* (EMC).

Definition 2.3.3 (Embedded Markov chain)**.** Given a continuous-time Markov chain $\{X(t) : t \geq 0\}$, the discrete-time Markov chain $\{Y_n : n \in \mathbb{N}\}$, where Y_n is the nth state visited by $X(t)$, is the *embedded Markov chain* for $X(t)$.

In particular, a CTMC is irreducible if and only if its EMC is irreducible. We note that there is no concept of periodicity for continuous-time Markov chains because there are no time steps at which transitions either do or do not occur [110]. The transition probabilities w_{ij} of the embedded Markov chain that corresponds to an irreducible CTMC are given by

$$w_{ij} = \begin{cases} -q_{ij}/q_{ii} & \text{if } i \neq j, \\ 0 & \text{if } i = j. \end{cases} \tag{2.12}$$

We note that irreducibility of the underlying continuous-time Markov chain implies that $q_{ii} \neq 0$ for all states i. By the definition of the transition rates it follows that

$$0 \leq w_{ij} \leq 1 \quad \text{for} \quad i, j = 1, \ldots, N \quad \text{and} \quad \sum_{i=1}^{N} w_{ij} = 1 \quad \text{for} \quad j = 1, \ldots, N.$$

Moreover, because $w_{ij} = 0$ if and only if $q_{ij} = 0$ for $i \neq j$, irreducibility of $\mathbf{Q}$ implies irreducibility of $\mathbf{W}$. Therefore, the transition probability matrix of the embedded Markov chain is an irreducible stochastic matrix. In matrix form

$$\mathbf{W} = \mathbf{I} - \mathbf{Q}(\mathrm{diag}(\mathbf{Q}))^{-1}, \tag{2.13}$$

where $\mathrm{diag}(\mathbf{Q})$ is the diagonal matrix corresponding to the diagonal of $\mathbf{Q}$.

Similar to DTMCs we can define probability distributions on the states of continuous-time Markov chains. We let $v_i(t)$ denote the probability of being in state i at time t, and define the column-vector $(\mathbf{v}(t))_i = v_i(t)$. Given some initial distribution $\mathbf{v}(0) = \mathbf{v}_0$ a time $t = 0$, the distribution at any future time $t > 0$ is a solution the following system of ordinary differential equations:

$$\frac{d\mathbf{v}(t)}{dt} = \mathbf{Q}\mathbf{v}(t),\ \mathbf{v}(0) = \mathbf{v}_0 \quad \Rightarrow \quad \mathbf{v}(t) = \exp(t\mathbf{Q})\mathbf{v}_0. \tag{2.14}$$

If there reaches a point in time at which the rate of change of $\mathbf{v}(t)$ is zero, then $d\mathbf{v}(t)/dt = 0$ and the system has reached a limiting distribution. For a finite irreducible CTMC the limiting distribution always exists and is identical to the stationary distribution. The limiting distribution, denoted by $\mathbf{v}$, satisfies

$$\mathbf{Q}\mathbf{v} = \mathbf{0}. \tag{2.15}$$

Therefore, the limiting distribution can be computed by applying linear system solvers directly to (2.15). Alternatively, the limiting distribution can be obtained from the stationary distribution of the corresponding embedded Markov chain. Let $\mathbf{W}$ be the transition matrix of the EMC (2.13), and let $\mathbf{u}$ denote its unique stationary distribution. Then

$$\mathbf{W}\mathbf{u} = \mathbf{u} \quad \Rightarrow \quad \mathbf{Q}(\mathrm{diag}(\mathbf{Q}))^{-1}\mathbf{u} = \mathbf{0} \quad \Rightarrow \quad \mathbf{v} = -\frac{(\mathrm{diag}(\mathbf{Q}))^{-1}\mathbf{u}}{\|(\mathrm{diag}(\mathbf{Q}))^{-1}\mathbf{u}\|_1}. \tag{2.16}$$

Since the vector space $\mathcal{Q} = \{\mathbf{v} : \mathbf{Q}\mathbf{v} = \mathbf{0}\}$ is isomorphic to $\mathcal{W} = \{\mathbf{u} : \mathbf{W}\mathbf{u} = \mathbf{u}\}$, it follows that (2.15) has a unique solution up to scalar multiplication. Therefore, $\mathbf{v}$ as given in (2.16) must be the unique limiting probability distribution. From a practical point of view, using the EMC has the potential drawback that $\mathbf{W}$ may be periodic. To avoid periodicity we can instead compute the stationary distribution of an alternative embedded process referred to as the *uniformized chain*. The uniformized chain is a discrete-time Markov chain that can be interpreted as a discretized version of the original CTMC, where transitions take place at intervals of length α. The transition matrix of the uniformized chain is given by

$$\mathbf{P} = \mathbf{I} + \alpha\mathbf{Q} \quad \text{for} \quad 0 < \alpha \leq (\max_i |q_{ii}|)^{-1}, \tag{2.17}$$

where the constraints on α ensure that $\mathbf{P}$ is a stochastic matrix. We note that $\mathbf{P}$ is irreducible if and only if $\mathbf{Q}$ is irreducible. Clearly, the stationary distribution of the uniformized chain is the same as the stationary distribution of the corresponding CTMC. If $\mathbf{P}$ is irreducible, and if $p_{ii} > 0$ for some i, it follows by Theorem 2.2.3 (4) that $\mathbf{P}$ is primitive and hence aperiodic. Thus, the uniformized chain corresponding to an irreducible CTMC is aperiodic for any $0 < \alpha < (\max_i |q_{ii}|)^{-1}$. We observe that as $\alpha \to 0$ all the eigenvalues of $\mathbf{P}$ converge to unity. Therefore, a good heuristic is to choose the parameter α as large as possible so as to try and maximize the separation between $\lambda_1 = 1$ and the *subdominant eigenvalues*, that is, the eigenvalues with modulus closest to but strictly less than one. In practice, maximizing the distance between the dominant and subdominant eigenvalues typically improves convergence of basic one-level iterative methods for linear systems.

The problem of computing the stationary distribution of an irreducible finite Markov chain can now be succinctly stated as follows.

Proposition 2.3.1. *(Stationary distribution of a Markov chain) Let $\{X(t) : t \in \mathfrak{T}\}$ be an irreducible Markov chain with a finite state space. If X is a DTMC let $\mathbf{P}$ be its transition probability matrix, otherwise let $\mathbf{P}$ be the transition matrix of the uniformized chain* (2.17)*. Then X has a unique stationary distribution that is the solution of*

$$(\mathbf{I} - \mathbf{P})\mathbf{x} = \mathbf{0}, \quad x_i > 0 \ \textit{for all} \ i, \quad \sum_i x_i = 1,$$

where $\mathbf{I} - \mathbf{P}$ is an irreducible singular M-matrix.

Proof. Since $\mathbf{P}$ is a stochastic matrix, $\rho(\mathbf{P}) = 1$, and hence $\mathbf{I} - \mathbf{P}$ is a singular M-matrix. Irreducibility follows by the fact that subtracting $\mathbf{P}$ from the identity matrix does not alter any of its off-diagonal elements. The claim now follows by Theorem 2.2.6 (1). ☐

We conclude this section by listing some properties of the matrix $\mathbf{A} = \mathbf{I} - \mathbf{P}$ that are required later in this thesis.

1. The matrix $\mathbf{A}$ is an irreducible singular M-matrix.
2. Each column of $\mathbf{A}$ sums to zero.
3. By Theorem 2.2.6 (4) the diagonal elements of $\mathbf{A}$ are strictly positive.
4. By Theorem 2.2.6 (5) any nonzero eigenvalue of $\mathbf{A}$ belongs to the disk of radius one centered at the point $(1, 0)$ in the complex plane.

2.4 Tensors

An Nth-order tensor is an N-dimensional array of size $I_1 \times \cdots \times I_N$. The *order* of a tensor is the number of modes (dimensions), and the size of the nth mode is I_n for $n = 1, \ldots, N$. Vectors are tensors of order one and matrices are tensors of order two. Tensors of order three and higher are referred to as higher-order tensors. Figure 2.3 illustrates a tensor of order three. An Nth-order tensor is *rank one* if it can be written as the outer product of

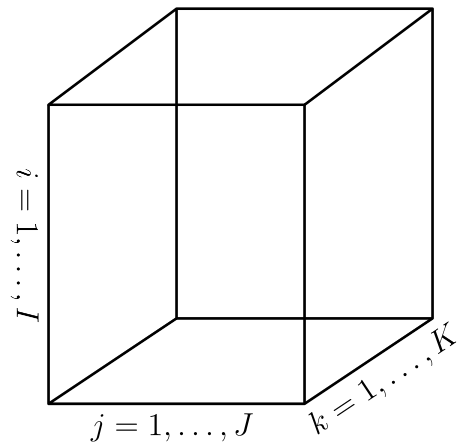


Figure 2.3: A third-order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

N vectors, that is,

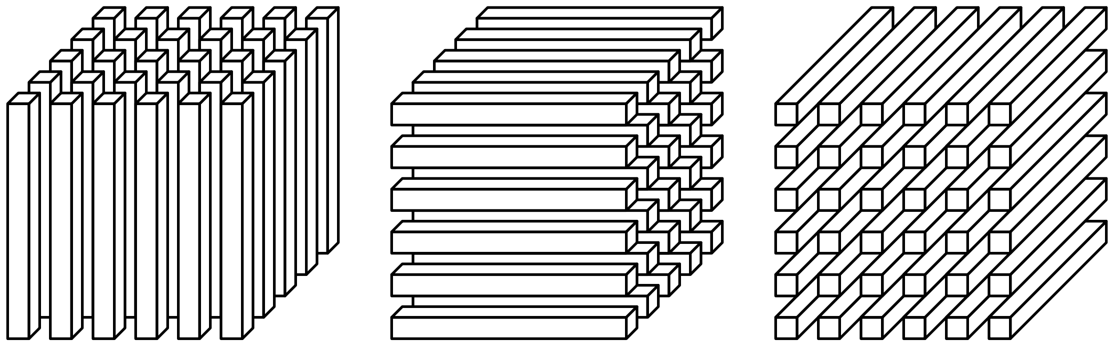
$$\mathcal{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(N)},$$

with the elements of $\mathcal{X}$ given by the corresponding product of vector elements:

$$x_{i_1,i_2,\ldots,i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2}, \ldots, a^{(N)}_{i_N} \quad \text{for all} \quad 1 \leq i_n \leq I_n.$$

In general, a tensor is rank R if it can be expressed exactly as a sum of no fewer than R rank one tensors. We note that this definition of tensor rank is one of many that exist in the literature; for other concepts of tensor rank see [79].

Fibers are higher-order generalizations of matrix rows and columns. A fiber is obtained by fixing every index of a tensor but one. Figure 2.4 illustrates column, row and tube fibers of a third-order tensor that are obtained by fixing index i, j, and k, respectively. Analogously, *slices* are two dimensional sections of a tensor, that are obtained by fixing all



(a) Mode-1 (column) fibers (b) Mode-2 (row) fibers (c) Mode-3 (tube) fibers

Figure 2.4: Fibers of a third-order tensor.

but two indices.

Matricization, also referred to as unfolding or flattening, is the process of reordering the elements of a tensor into a matrix. In this thesis we are only interested in mode-n matricization, which arranges the mode-n fibers to be the columns of the resulting matrix. The mode-n matricized version of a tensor $\mathcal{Z}$ is denoted by $\mathbf{Z}_{(n)}$. Matricization provides an elegant way to describe the product of a tensor by a matrix in mode n. The n-mode matrix product of a tensor $\mathcal{Z} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{Z} \times_n \mathbf{A}$ and is of size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$. The n-mode product can be expressed in terms of unfolded tensors as follows:

$$\mathcal{X} = \mathcal{Z} \times_n \mathbf{A} \quad \Leftrightarrow \quad \mathbf{X}_{(n)} = \mathbf{A}\mathbf{Z}_{(n)}.$$

We note that in a sequence of multiplications the order of multiplication is irrelevant for distinct modes in the sequence, that is,

$$\mathcal{Z} \times_n \mathbf{A} \times_m \mathbf{B} = \mathcal{Z} \times_n \mathbf{B} \times_m \mathbf{A} \ \text{ for } \ m \neq n.$$

If the modes are the same, then

$$\mathcal{Z} \times_n \mathbf{A} \times_n \mathbf{B} = \mathcal{Z} \times_n (\mathbf{B}\mathbf{A}).$$

To denote the product of a tensor and a sequence of matrices over some nonempty subset of the modes $\mathcal{N} = \{n_1, \ldots, n_k\} \subset \{1, \ldots, N\}$, we use the following notation as shorthand

$$\mathcal{Z} \times_{n \in \mathcal{N}} \mathbf{A}^{(n)} = \mathcal{Z} \times_{n_1} \mathbf{A}^{(n_1)} \cdots \times_{n_k} \mathbf{A}^{(n_k)}.$$

The *Kronecker product* of two matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times L}$ is denoted by $\mathbf{A} \otimes \mathbf{B}$. The result is a matrix of size $IJ \times KL$ given by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix}.$$

The Kronecker product is a bilinear and associative operation that satisfies the following *mixed-product property*:

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$$

for any matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ of the appropriate sizes. By the mixed-product property it follows that $(\mathbf{A} \otimes \mathbf{B})$ is invertible if and only if $\mathbf{A}$ and $\mathbf{B}$ are invertible, in which case the inverse is given by

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}.$$

Similarly, the transpose operation is distributive over the Kronecker product

$$(\mathbf{A} \otimes \mathbf{B})^{\top} = \mathbf{A}^{\top} \otimes \mathbf{B}^{\top}.$$

Armed with the Kronecker product we can now state the following useful relationship between tensors and their matricized versions [78]. Let $\mathbfcal{Z} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\mathbf{A}^{(n)} \in \mathbb{R}^{J_n \times I_n}$ for $n = 1, \ldots, N$. Then, for any $n \in \{1, \ldots, N\}$,

$$\begin{aligned} &\mathbfcal{X} = \mathbfcal{Z} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)} \\ &\Updownarrow \\ &\mathbf{X}_{(n)} = \mathbf{A}^{(n)} \mathbf{Z}_{(n)} \left(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)} \right)^{\top}. \end{aligned} \tag{2.18}$$

The *Khatri–Rao product* of two matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$ is a matrix of size $(IJ) \times K$ given by

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \;\; \mathbf{a}_2 \otimes \mathbf{b}_2 \;\; \cdots \;\; \mathbf{a}_K \otimes \mathbf{b}_K].$$

The Khatri–Rao product is a bilinear and associative operation that is equivalent to the columnwise Kronecker product. Associativity of the Khatri–Rao and Kronecker products, and the mixed-product property of the Kronecker product imply the following useful result:

$$\mathbf{A}^{(1)}\mathbf{B}^{(1)} \odot \cdots \odot \mathbf{A}^{(N)}\mathbf{B}^{(N)} = \left(\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)}\right) \left(\mathbf{B}^{(1)} \odot \cdots \odot \mathbf{B}^{(N)}\right) \tag{2.19}$$

for any sequences of matrices $\mathbf{A}^{(n)}$ and $\mathbf{B}^{(n)}$ of the appropriate sizes.

The *Hadamard product* is the elementwise matrix product. Given matrices $\mathbf{A}$ and $\mathbf{B}$

both of size $I \times J$, the Hadamard product of $\mathbf{A}$ and $\mathbf{B}$ is the $I \times J$ matrix given by

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix}.$$

The Hadamard product is associative, commutative, and distributive over addition. The Hadamard product also preserves symmetry in that $\mathbf{A} * \mathbf{B}$ is symmetric if $\mathbf{A}$ and $\mathbf{B}$ are symmetric. We note that the Hadamard product of two matrices may be symmetric even if the individual matrices are not.

2.5 Stationary iterative methods

In preparation for an introduction to algebraic multigrid, we first consider simple stationary iterative methods for solving the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{f} \tag{2.20}$$

with $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{x}$, $\mathbf{f} \in \mathbb{R}^n$. These methods play an important role in multigrid where they typically compose the *relaxation scheme* or *smoother* of the multigrid solver. The purpose of the relaxation scheme is to eliminate error in the approximate solution, and the components of the error that are not effectively reduced by relaxation are called the *algebraically smooth error*. The smooth error components must then be eliminated through a complementary process referred to as the *coarse-grid correction* procedure. However, we defer any further discussion of algebraic multigrid to §2.6.3, where these concepts are explored in detail. The remainder of this section introduces iterative methods based on matrix splittings such as the Jacobi and Gauss–Seidel iterations, and states conditions for their convergence. In particular, the application of these methods to irreducible singular M-matrices is investigated. We recommend [11] and [126] for an in-depth exploration of the convergence of iterative methods based on matrix splittings.

In general, given a splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$ in which $\mathbf{M}$ is nonsingular, we seek a solution of (2.20) by way of the following stationary iterative procedure

$$\mathbf{x}^{(k+1)} = \mathbf{M}^{-1}\mathbf{N}\mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{f} = \mathbf{H}\mathbf{x}^{(k)} + \mathbf{c} \tag{2.21}$$

for $k = 0, 1, 2, \ldots$ and some initial guess $\mathbf{x}^{(0)}$. The matrix $\mathbf{H}$ is referred to as the *iteration matrix* for the iterative procedure. The qualifier *stationary* refers to the fact that the update formula does not change from one iteration to the next. We note that since $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$, the iterative procedure (2.21) can also be written as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{r}^{(k)} \tag{2.22}$$

where $\mathbf{r}^{(k)} = \mathbf{f} - \mathbf{A}\mathbf{x}^{(k)}$ is the residual of the kth approximation. Therefore, the new approximation is obtained from the current one by adding a transformed residual. The iterative procedure in (2.21) can also be viewed as a technique for solving the system

$$(\mathbf{I} - \mathbf{H})\mathbf{x} = \mathbf{c}. \tag{2.23}$$

Writing the iteration matrix as $\mathbf{H} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$, this system can be rewritten as

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{f}. \tag{2.24}$$

The above system, which has the same solutions as the original system (2.20), is the preconditioned system, where $\mathbf{M}$ is the *preconditioning matrix*. Evidently, if $\mathbf{M}^{-1}$ is a good approximation of $\mathbf{A}^{-1}$ (assuming that $\mathbf{A}$ is invertible) then we expect that an iterative method applied to (2.24) would converge to the exact solution in only a few iterations. Therefore, the splitting should ideally be chosen such that $\mathbf{M}^{-1}$ approximates $\mathbf{A}^{-1}$ in some sense, and the inverse of $\mathbf{M}$ is cheap to compute.

A simple way to choose a splitting is to decompose the coefficient matrix $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$, where $\mathbf{D} = \operatorname{diag}(\mathbf{A})$, and $\mathbf{L}$ and $\mathbf{U}$ are respectively the lower and upper triangular parts of $\mathbf{A}$ with their signs reversed and with zeros on their diagonals. The (pointwise) Jacobi and

Gauss–Seidel (GS) iterations then correspond to the following splittings

$$\text{Jacobi: } \mathbf{M} = \mathbf{D}, \quad \mathbf{N} = \mathbf{L} + \mathbf{U}, \quad \mathbf{H}_J = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}) \tag{2.25}$$

$$\text{Gauss–Seidel: } \mathbf{M} = \mathbf{D} - \mathbf{L}, \quad \mathbf{N} = \mathbf{U}, \quad \mathbf{H}_{GS} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}. \tag{2.26}$$

The Jacobi and Gauss–Seidel iterations are well-defined only if $\mathbf{A}$ has nonzero elements on its diagonal, otherwise $\mathbf{M}$ is not invertible. Considering the pointwise update equations the Jacobi and Gauss–Seidel iterations:

$$\text{Jacobi: } d_{ii}x_i^{(k+1)} = f_i - \sum_{j \neq i}(\ell_{ij} + u_{ij})x_j^{(k)} \tag{2.27}$$

$$\text{Gauss–Seidel: } d_{ii}x_i^{(k+1)} = f_i - \sum_{j<i} \ell_{ij}x_j^{(k+1)} - \sum_{j>i} u_{ij}x_j^{(k)}, \tag{2.28}$$

we observe that Gauss–Seidel uses the most recent information as soon as it is available, whereas Jacobi performs an entire iteration before using the updated values. Therefore, we might expect Gauss–Seidel to converge faster Jacobi, which in general it does. Also, from a practical viewpoint, Gauss–Seidel can be implemented using only a single array to hold the current approximation, which is simply overwritten during each iteration. The memory requirements for Jacobi, however, are higher because two arrays must be maintained to house the current and updated approximations. This is not to say the Jacobi is without merit, as it does possess benefits over Gauss–Seidel such as better parallelization properties [119] and a cheaper per iteration cost. Weighted versions of the Jacobi and Gauss–Seidel methods that depend on a relaxation parameter ω can also be defined. For example, the successive overrelaxation (SOR) method, which is a weighted variant of the Gauss–Seidel method, corresponds to the splitting

$$\mathbf{A} = \frac{1}{\omega}(\mathbf{D} - \omega\mathbf{L}) - \frac{1}{\omega}((1-\omega)\mathbf{D} + \omega\mathbf{U}).$$

In general, the method is overcorrecting when $\omega > 1$, and is undercorrecting when $\omega < 1$. When $\omega = 1$ we recover the unweighted version of the iterative method. The iteration matrices for weighted Jacobi and SOR are given by

$$\text{weighted Jacobi: } \mathbf{H}_{\omega J} = (1-\omega)\mathbf{I} + \omega \mathbf{H}_J = \mathbf{I} - \omega \mathbf{D}^{-1}\mathbf{A} \tag{2.29}$$

$$\text{SOR: } \mathbf{H}_{SOR} = (\mathbf{D} - \omega \mathbf{L})^{-1}[(1-\omega)\mathbf{D} + \omega \mathbf{U}]. \tag{2.30}$$

Adding a relaxation parameter to a stationary iterative method can often improve its convergence properties. For example, the convergence rate of SOR with the optimal value of ω can be a considerable improvement over that of Gauss–Seidel. Moreover, using a relaxation parameter can sometimes be the difference between a method converging or not converging, as we shall soon see. Unfortunately, an analytical determination of the optimal value of ω (or even a reasonable value) may require a fairly sophisticated eigenvalue analysis. While results do exist for certain classes of matrices (see Chapter 7 in [11] and [126]), little is currently known for general nonsymmetric matrices. However, it has been established that SOR converges for any matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ only if $\omega \in (0,2)$ (see Theorem 7.4.5 in [11])

So far we have introduced the general linear stationary iterative procedure based on a matrix splitting (2.21), and have mentioned some of the prevailing iterative methods that are used today. We now discuss conditions for which linear stationary iterative methods converge. We begin with some general convergence results, and then consider the case of (irreducible) singular M-matrices.

The first question we ask is: if the iteration $\mathbf{x}^{(k+1)} = \mathbf{H}\mathbf{x}^{(k)} + \mathbf{c}$ converges, then is the limit a solution of the original system? If the iteration sequence $\{\mathbf{x}^{(k)}\}$ converges to a limit $\mathbf{x}$, then its limit $\mathbf{x}$ satisfies

$$\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{c} \quad \Leftrightarrow \quad \mathbf{M}\mathbf{x} = \mathbf{N}\mathbf{x} + \mathbf{f} \quad \Leftrightarrow \quad \mathbf{A}\mathbf{x} = \mathbf{f}. \tag{2.31}$$

Hence, the limit of the iterative procedure is indeed a solution of the original system. A classical convergence result that applies to nonsingular linear systems is given below by Theorem 2.5.1 (see [104]).

Theorem 2.5.1. *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *be nonsingular. Then the iterative method* (2.21) *converges to the solution of* $\mathbf{Ax} = \mathbf{f}$ *for any initial guess* $\mathbf{x}^{(0)}$ *if* $\rho(\mathbf{H}) < 1$.

Proof. Let $\mathbf{x}^{(0)} \in \mathbb{R}^n$, and let $\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$ denote the error of the kth iterate $\mathbf{x}^{(k)}$ produced by the iterative method (2.21). Since $\mathbf{Hx} = \mathbf{x} - \mathbf{M}^{-1}\mathbf{f}$, it follows that

$$\mathbf{e}^{(k+1)} = \mathbf{He}^{(k)} = \mathbf{H}^{k+1}\mathbf{e}^{(0)} \quad \text{for} \quad k = 0, 1, 2, \ldots .$$

If $\rho(\mathbf{H}) < 1$, then the limit $\lim_{k\to\infty} \mathbf{H}^k$ exists and is equal to the zero matrix[1], i.e., $\mathbf{e}^{(k)} \to \mathbf{0}$ in the limit as $k \to \infty$. Therefore, the iterative procedure converges to the exact solution $\mathbf{x}$ for any initial guess $\mathbf{x}^{(0)}$. □

The asymptotic rate at which the iteration converges is given by the *global asymptotic convergence factor* ρ, defined by

$$\rho := \lim_{k\to\infty}\left(\max_{\mathbf{x}^{(0)}\in\mathbb{R}^n} \frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{e}^{(0)}\|}\right)^{1/k} = \lim_{k\to\infty}\left(\max_{\mathbf{e}^{(0)}\in\mathbb{R}^n} \frac{\|\mathbf{H}^k\mathbf{e}^{(0)}\|}{\|\mathbf{e}^{(0)}\|}\right)^{1/k} = \lim_{k\to\infty} \|\mathbf{H}^k\|^{1/k} = \rho(\mathbf{H}). \quad (2.32)$$

Thus, the rate of convergence of a stationary iterative method is governed by the spectral radius of the iteration matrix (when the coefficient matrix is nonsingular). In general, the smaller the spectral radius of $\mathbf{H}$, the faster the convergence of the iteration.

Since the spectral radius is bounded above by the norm of the matrix, the implication of Theorem 2.5.1 remains true under the more restrictive condition $\|\mathbf{H}\| < 1$, where $\|\cdot\|$ is any matrix norm. However, computing the spectral radius is typically expensive, and because a tight upper bound on $\rho(\mathbf{H})$ may be unattainable, convergence conditions based on properties of the splitting and the coefficient matrix $\mathbf{A}$ instead of $\rho(\mathbf{H})$ are convenient. Subsequent results in this section rely on the notion of regular and weak regular splittings.

Definition 2.5.1 (Regular and weak regular splitting)**.** A splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$ is called a *regular splitting* if the matrices $\mathbf{M}^{-1}$ and $\mathbf{N}$ are nonnegative. It is called a *weak regular splitting* if $\mathbf{M}^{-1}$ and $\mathbf{M}^{-1}\mathbf{N}$ are nonnegative.

[1]In this case the matrix $\mathbf{H}$ is said to be *convergent* (see [11] and Theorem 5.6.12 in [68]).

Theorem 2.5.2. *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *be an M-matrix (singular or nonsingular) with positive elements on its diagonal. Then weighted Jacobi and SOR correspond to regular splittings of* $\mathbf{A}$ *for* $0 < \omega \leq 1$.

Proof. First consider weighted Jacobi. We have that

$$\mathbf{M} = \frac{1}{\omega}\mathbf{D}, \quad \mathbf{N} = (1/\omega - 1)\mathbf{D} + \mathbf{L} + \mathbf{U},$$

which is clearly a regular splitting if $\omega > 0$ and $1/\omega - 1 \geq 0$, i.e., if $0 \leq \omega < 1$. Now consider the SOR method. Without loss of generality suppose that $\mathbf{A}$ has a unit diagonal. Then we have that

$$\mathbf{M} = \frac{1}{\omega}(\mathbf{I} - \omega\mathbf{L}), \quad \mathbf{N} = \frac{1}{\omega}[(1-\omega)\mathbf{I} + \omega\mathbf{U}].$$

Clearly $\mathbf{N}$ is nonnegative if $0 < \omega \leq 1$. Since $\mathbf{M}$ is nonsingular, and since $\mathbf{L}$ is strictly lower triangular and nonnegative, it follows that

$$\mathbf{M}^{-1} = \omega(\mathbf{I} - \omega\mathbf{L})^{-1} = \omega(\mathbf{I} + \omega\mathbf{L} + \omega^2\mathbf{L}^2 + \ldots + \omega^{n-1}\mathbf{L}^{n-1}),$$

which is nonnegative if $\omega > 0$. Hence, for $0 < \omega \leq 1$, SOR is based on a regular splitting. □

The following result (Theorem 4.4 in [104]) applies to regular splittings of nonsingular inverse-positive matrices.

Theorem 2.5.3. *Let* $\mathbf{A} = \mathbf{M} - \mathbf{N}$ *be a regular splitting. Then* $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$ *if and only if* $\mathbf{A}$ *is nonsingular and* $\mathbf{A}^{-1}$ *is nonnegative.*

Proof. We begin with the sufficient condition. Writing

$$\mathbf{A} = \mathbf{M}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{N}),$$

it is obvious that $\mathbf{A}$ is nonsingular if $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$. To show that $\mathbf{A}$ is inverse positive we observe that

$$\mathbf{A}^{-1} = (\mathbf{I} - \mathbf{M}^{-1}\mathbf{N})^{-1}\mathbf{M}^{-1} = \left[\mathbf{I} + \sum_{n=1}^{\infty}(\mathbf{M}^{-1}\mathbf{N})^{n}\right]\mathbf{M}^{-1}.$$

Since $\mathbf{M}$, $\mathbf{N}$ is a regular splitting of $\mathbf{A}$, this expression consists of an infinite sum of nonnegative terms, hence $\mathbf{A}^{-1}$ must be nonnegative.

Now consider the necessary condition. Assuming that $\mathbf{A}$ is nonsingular, it follows that $\mathbf{I} - \mathbf{M}^{-1}\mathbf{N}$ is nonsingular. Therefore, we are justified in writing

$$\mathbf{A}^{-1}\mathbf{N} = (\mathbf{I} - \mathbf{H})^{-1}\mathbf{H},$$

where $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N}$. Since $\mathbf{H}$ is nonnegative, by the Perron–Frobenius theorem there exists a nonnegative eigenvector $\mathbf{x}$ of $\mathbf{H}$ with corresponding eigenvalue $\rho(\mathbf{H})$. Therefore,

$$\mathbf{A}^{-1}\mathbf{N}\mathbf{x} = (\mathbf{I} - \mathbf{H})^{-1}\mathbf{H}\mathbf{x} = \frac{\rho(\mathbf{H})}{1 - \rho(\mathbf{H})}\mathbf{x}.$$

Using the fact that $\mathbf{A}^{-1}$, $\mathbf{N}$, and $\mathbf{x}$ are nonnegative, it follows that

$$\frac{\rho(\mathbf{H})}{1 - \rho(\mathbf{H})} \geq 0,$$

which is only possible if $0 \leq \rho(\mathbf{H}) \leq 1$. However, because $\mathbf{I} - \mathbf{H}$ is nonsingular it must be true that $\rho(\mathbf{H}) \neq 1$, which leads us to conclude that $\rho(\mathbf{M}^{-1}\mathbf{N}) = \rho(\mathbf{H}) < 1$.

$\square$

By Theorem 2.2.7 (3), a corollary of this result is that the iterative method (2.21) converges for any regular splitting of a nonsingular M-matrix. In particular, by Theorem 2.5.2, weighted Jacobi and SOR converge for $0 < \omega \leq 1$ when the coefficient matrix is a nonsingular M-matrix.

We now consider convergence results for the iterative method (2.21) in which the coefficient matrix $\mathbf{A}$ is singular. It is assumed that the linear system $\mathbf{A}\mathbf{x} = \mathbf{f}$ is consistent, that is, $\mathbf{f} \in \text{range}(\mathbf{A})$. Because $(1, \mathbf{x})$ with $\mathbf{x} \in \text{null}(\mathbf{A}) \setminus \{\mathbf{0}\}$ is an eigenpair of $\mathbf{H} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$ for any invertible matrix $\mathbf{M}$, it follows that $\rho(\mathbf{H}) \geq 1$. In this case the iteration matrix cannot be convergent, and we require a weaker condition on $\mathbf{H}$, namely, semiconvergence.

Definition 2.5.2 (Semiconvergent matrix)**.** A matrix $\mathbf{H} \in \mathbb{R}^{n \times n}$ is said to be *semiconvergent* whenever the limit $\lim_{k\to\infty} \mathbf{H}^k$ exists.

The following theorem shows that semiconvergence is a necessary and sufficient condition for the iterative method (2.21) to converge (see Lemma 7.6.13 in [11]). We note that the matrix $\mathbf{A}$ in Theorem 2.5.4 may be either singular or nonsingular.

Theorem 2.5.4. *Let* $\mathbf{A} = \mathbf{M} - \mathbf{N} \in \mathbb{R}^{n \times n}$ *in which* $\mathbf{M}$ *is invertible. Then the iterative method* (2.21) *converges to a solution of* $\mathbf{A}\mathbf{x} = \mathbf{f}$ *for each initial guess* $\mathbf{x}^{(0)}$ *if and only if* $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N}$ *is semiconvergent.*

Necessary and sufficient conditions for a matrix to be semiconvergent are given below (see page 152 in [11]).

Theorem 2.5.5 (Conditions for semiconvergence)**.** *Let* $\mathbf{H} \in \mathbb{R}^{n \times n}$. *Then* $\mathbf{H}$ *is semiconvergent if and only if each of the following conditions hold.*

1. *The spectral radius* $\rho(\mathbf{H}) \leq 1$.
2. *If* $\rho(\mathbf{H}) = 1$ *then* $\text{rank}((\mathbf{I} - \mathbf{H})^2) = \text{rank}(\mathbf{I} - \mathbf{H})$.
3. *If* $\rho(\mathbf{H}) = 1$ *then* $\lambda \in \sigma(\mathbf{H})$ *with* $|\lambda| = 1$ *implies that* $\lambda = 1$.

It turns out that the first two conditions of Theorem 2.5.5 are satisfied by any regular splitting of an irreducible singular M-matrix (see Theorem 6.4.12 in [11]). Consequently, by Theorem 2.5.2 the Jacobi and Gauss–Seidel iteration matrices satisfy the first two conditions. However, the third condition, which says that any eigenvalue in $\sigma(\mathbf{H}) \setminus \{1\}$ must belong to the open unit disk in the complex plane, may not hold true. For example, although the transition matrix $\mathbf{P}$ of a Markov chain may be primitive, the iteration matrix corresponding to a regular splitting of $\mathbf{I} - \mathbf{P}$ may be periodic, in that it has multiple eigenvalues distributed uniformly along the boundary of the unit circle. Consequently, the Jacobi and Gauss–Seidel iterations are not guaranteed to converge. Now consider the weighted Jacobi method applied to an irreducible singular M-matrix. As shown in (2.30), the weighted Jacobi iteration matrix can be written in terms of the Jacobi iteration matrix as follows

$$\mathbf{H}_{\omega J} = (1-\omega)\mathbf{I} + \omega \mathbf{H}_J.$$

Consequently, any eigenvalue λ of $\mathbf{H}_{\omega J}$ is given by $\lambda = 1-\omega+\omega\mu$ for some $\mu \in \sigma(\mathbf{H}_J)$. A little algebra shows that

$$|\lambda - (1-\omega)| = \omega|\mu| \leq \omega \quad \text{for all} \quad \lambda \in \sigma(\mathbf{H}_{\omega J}).$$

Therefore, if $0 < \omega < 1$ then $\lambda = 1$ is the only eigenvalue of $\mathbf{H}_{\omega J}$ that lies on the unit circle, and hence, weighted Jacobi applied to an irreducible singular M-matrix converges for any $\omega \in (0,1)$. As illustrated in Figure 2.5, the relaxation parameter ω serves to shrink $\sigma(\mathbf{H})$ away from the boundary of the unit circle toward the unit eigenvalue.

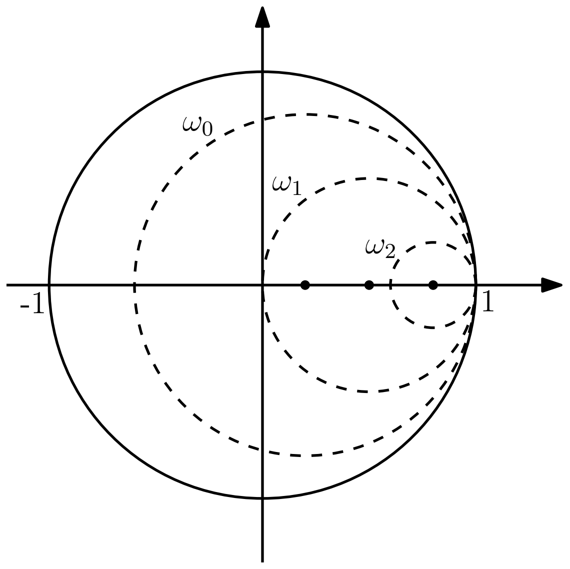


Figure 2.5: Disks $\{\lambda \in \mathbb{C} : |\lambda - (1-\omega)| \leq \omega\}$ that contain the spectrum of $\mathbf{H}_{\omega J}$ for $0 < \omega_2 < \omega_1 < \omega_0 < 1$. The solid dots indicate the centers of the disks.

Let us consider the role of the relaxation parameter a bit further. The *asymptotic rate of convergence* of the iterative procedure (2.21) with semiconvergent iteration matrix $\mathbf{H}$ is defined as

$$R_\infty(\mathbf{H}) = -\ln \delta(\mathbf{H}) \quad \text{with} \quad \delta(\mathbf{H}) = \max\{|\lambda| : \lambda \in \sigma(\mathbf{H}),\ \lambda \neq 1\}. \tag{2.33}$$

The larger the value of $R_\infty(\mathbf{H})$, or equivalently the smaller the value of $\delta(\mathbf{H})$, the faster the convergence of the iterative process. The quantity $\delta(\mathbf{H})$, which is strictly less than $\rho(\mathbf{H})$ when the coefficient matrix is singular, corresponds to the magnitude of the subdominant eigenvalue(s) of $\mathbf{H}$. Therefore, the optimal parameter ω is one that minimizes the magnitude of the subdominant eigenvalue(s) of $\mathbf{H}_{\omega J}$. We now ask ourselves the following question. Given that $\lambda = 1-\omega+\omega\mu$ for $\mu \in \sigma(\mathbf{H}_J)$ is an eigenvalue of $\mathbf{H}_{\omega J}$, to what degree can we minimize $|\lambda|$ by our choice of ω? To answer this question we consider a contour plot (Figure 2.6) of the function

$$f(\mu) = |1 - \omega_{opt}(\mu) + \omega_{opt}(\mu)\mu|$$

for all $\mu \in \{z \in \mathbb{C} : |z| < 1\}$, where

$$\omega_{opt}(\mu) = \min\left\{0.99, \frac{1 - \mathrm{Re}(\mu)}{(1 - \mathrm{Re}(\mu))^2 + \mathrm{Im}(\mu)^2}\right\}$$

is the value of $\omega \in (0, 1)$ that minimizes $f(\mu)$. Because the true optimal value of ω may be greater than one, we take the minimum with respect to 0.99 in order to restrict ω to the interval $(0, 1)$. We note that our choice of 0.99 is arbitrary, and any number sufficiently close to but strictly less than one would have sufficed. Figure 2.6 shows that the optimal relaxation parameter provides effective damping when $\mathrm{Re}(\mu) < 0$. However, we also observe that the effectiveness of the damping decreases as μ approaches one, and in particular, the relaxation parameter has little effect when $\mathrm{Re}(\mu) \approx 1$. Consequently, if $\mathbf{H}_J$ has eigenvalues with real part close to one we expect the weighted Jacobi method to be slow to converge regardless of the relaxation parameter (note that $\mathrm{Re}(\mu) \approx 1 \Rightarrow \mathrm{Re}(\lambda) \approx 1$).

The Gauss–Seidel method can also be made to converge by using a weighted iteration

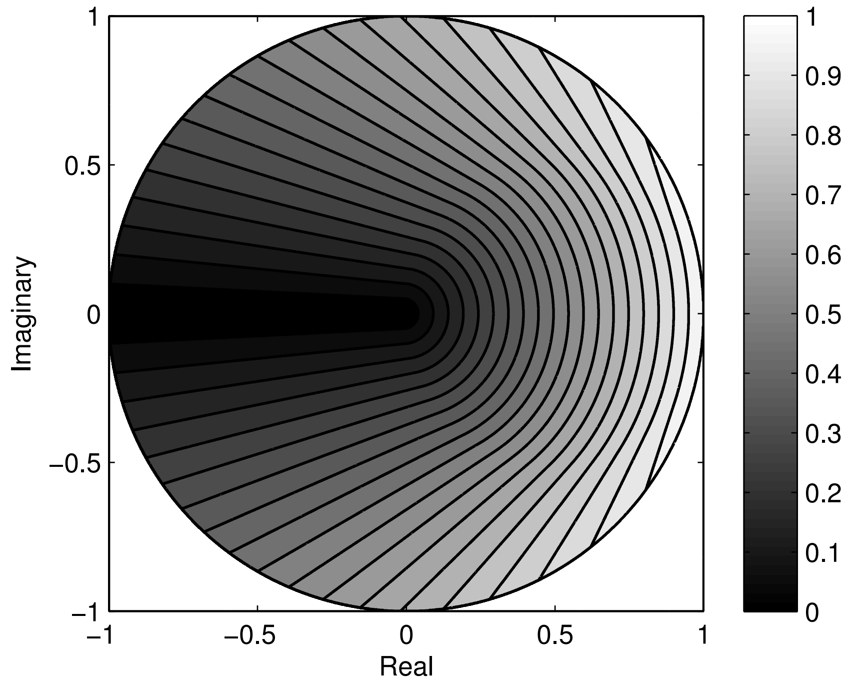


Figure 2.6: Contour plot of $f(\mu)$ over the open unit disk in the complex plane.

matrix of the form

$$\mathbf{H}_{\omega GS} = (1-\omega)\mathbf{I} + \omega\mathbf{H}_{GS} = \mathbf{I} - \omega(\mathbf{D}-\mathbf{L})^{-1}\mathbf{A}.$$

It must be stressed, however, that while the weighted Gauss–Seidel method is similar to the SOR method, they are not the same. In general, if $\mathbf{H}$ is the iteration matrix arising from a weak regular splitting of an irreducible singular M-matrix, then

$$\mathbf{H}_{\omega} = (1-\omega)\mathbf{I} + \omega\mathbf{H} \tag{2.34}$$

is semiconvergent for any $\omega \in (0,1)$ [93].

We conclude this section by showing that the weighted Jacobi method with $\omega \in (0,1)$ is guaranteed to converge to the unique (up to scaling) strictly positive solution of $\mathbf{Ax} = \mathbf{0}$ when $\mathbf{A}$ is an irreducible singular M-matrix.

Theorem 2.5.6. *Consider the linear system* $\mathbf{Ax} = \mathbf{0}$ *where* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *is an irreducible singular M-matrix. Let* $\mathbf{A} = \mathbf{M} - \mathbf{N}$ *be any weak regular splitting chosen in such a way that* $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N}$ *is semiconvergent. Then the iterative procedure* (2.21) *converges to a strictly positive solution of* $\mathbf{Ax} = \mathbf{0}$ *for any strictly positive initial guess* $\mathbf{x}^{(0)}$*.*

Proof. The iterative procedure applied to the homogeneous system is given by

$$\mathbf{x}^{(k)} = \mathbf{H}^k\mathbf{x}^{(0)} \quad \text{for} \quad k = 1, 2, 3, \ldots.$$

By Theorem 2.5.4 this procedure converges to a solution of $\mathbf{Ax} = \mathbf{0}$. Therefore, our goal is to show that the solution is not the trivial solution. Let

$$\hat{\mathbf{H}} = \lim_{k\to\infty} \mathbf{H}^k,$$

which exists because $\mathbf{H}$ is semiconvergent. By the assumption of a weak regular splitting, $\mathbf{H}^k$ is nonnegative for any $k \in \mathbb{N}$, and hence $\hat{\mathbf{H}}$ is nonnegative. Let

$$\hat{\mathbf{x}} = \lim_{k\to\infty} \mathbf{x}^{(k)} = \lim_{k\to\infty} \mathbf{H}^k\mathbf{x}^{(0)} = \hat{\mathbf{H}}\mathbf{x}^{(0)}.$$

By Lemma 7.6.11 in [11]

$$\hat{\mathbf{H}} = \mathbf{I} - (\mathbf{I} - \mathbf{H})(\mathbf{I} - \mathbf{H})^{\mathrm{D}},$$

where $(\mathbf{I} - \mathbf{H})^{\mathrm{D}}$ is the unique matrix[2] such that for any $\mathbf{z} \in \mathbb{R}^n$

$$(\mathbf{I} - \mathbf{H})^{\mathrm{D}}\mathbf{z} = \begin{cases} \mathbf{y} & \text{if } (\mathbf{I} - \mathbf{H})\mathbf{y} = \mathbf{z} \text{ and } \mathbf{x} \in \operatorname{range}(\mathbf{I} - \mathbf{H}), \\ \mathbf{0} & \text{if } (\mathbf{I} - \mathbf{H})\mathbf{z} = \mathbf{0}. \end{cases}$$

Writing $\mathbf{H} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$, we have $\hat{\mathbf{H}} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{D}}$. Let $\mathbf{x}$ be the strictly positive vector that spans the one-dimensional nullspace of $\mathbf{A}$. Since $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{0}$ we have

$$\hat{\mathbf{H}}\mathbf{x} = [\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{D}}]\mathbf{x} = \mathbf{x}.$$

[2]The matrix $(\mathbf{I} - \mathbf{H})^{\mathrm{D}}$ is the *Drazin inverse* of $\mathbf{I} - \mathbf{H}$; see page 198 in [11].

Therefore, by strict positivity of the exact solution, $\hat{\mathbf{H}}$ must have at least one strictly positive element in each of its rows, otherwise, $\hat{\mathbf{H}}\mathbf{x}$ would have a zero component, which is not possible. Hence, by strict positivity of the initial guess and because $\dim \operatorname{null}(\mathbf{A}) = 1$, $\hat{\mathbf{x}}$ must be the desired strictly positive nontrivial solution. □

Corollary 2.5.1. *If* $\mathbf{A}$ *is an irreducible singular M-matrix, and if* $\mathbf{A} = \mathbf{M} - \mathbf{N}$ *is a weak regular splitting chosen in such a way that* $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N}$ *is semiconvergent, then the iterative procedure* (2.21) *applied to* $\mathbf{A}\mathbf{x} = \mathbf{0}$ *has strictly positive iterates.*

Proof. Because $\hat{\mathbf{H}}$ has at least one strictly positive element in each row, $\mathbf{H}^k$ must also have at least one strictly positive element in each row for any $k \in \mathbb{N}$. Therefore, given a strictly positive initial guess, the iterates of (2.21) must be strictly positive for all $k \in \mathbb{N}$. □

Theorem 2.5.7. *For any irreducible singular M-matrix* $\mathbf{A} \in \mathbb{R}^{n \times n}$*, the weighted Jacobi method with* $0 < \omega < 1$ *converges to the unique (up to scaling) strictly positive solution of* $\mathbf{A}\mathbf{x} = \mathbf{0}$ *for any strictly positive initial guess* $\mathbf{x}^{(0)}$*.*

Proof. By Theorem 2.2.6 (4) $\mathbf{A}$ has strictly positive diagonal elements. Therefore, the iteration matrix for the weighted Jacobi method is based on a regular splitting by Theorem 2.5.2 and is semiconvergent for $0 < \omega < 1$. The result now follows by Theorem 2.5.6. □

2.6 Multigrid

2.6.1 Principles of multigrid

Multigrid methods (we first focus on geometric multigrid) were originally developed to solve linear systems arising from the discretization of boundary value problems on spatial domains. Multigrid's development was motivated in part by the observation that while simple iterative methods such as Jacobi and Gauss–Seidel may be slow to converge for such problems, they possess a *smoothing property* in that the approximation error after a few iterations is geometrically smooth. The smoothing property of simple iterative methods, and the fact that smooth error can be well approximated on coarser grids, are the two basic principles of the multigrid approach. These concepts are best illustrated by considering the simple boundary value problem:

$$\begin{aligned} -u_{xx} - u_{yy} &= f(x,y) \quad \text{for} \quad (x,y) \in \Omega = (0,1) \times (0,1) \\ u(x,y) &= 0 \quad \text{for} \quad (x,y) \in \partial\Omega, \end{aligned} \tag{2.35}$$

which is the Poisson problem on the unit square with homogeneous Dirichlet boundary conditions for some function $f(x,y)$. Suppose that $\bar{\Omega} = \Omega \cup \partial\Omega$ is discretized by the grid points $(x_i, y_j) = (ih, jh)$ for $i, j = 0, \ldots, n$ and $n \in \mathbb{N}$, where for simplicity the grid spacing $h = h_x = h_y = 1/n$ is the same in both dimensions. The discrete grid including the boundary points is denoted by $\bar{\Omega}_h$, and the discrete grid corresponding to the interior points is denoted by Ω_h. Evaluating the exact solution $u(x,y)$ at the grid points in $\bar{\Omega}_h$ we define the grid function $u_{ij} = u(x_i, y_j)$. Similarly, we define $f_{ij} = f(x_i, y_j)$ at grid points in Ω_h. The partial derivatives in (2.35) can then be replaced by their centered finite difference approximations which gives rise to the discrete problem:

$$\begin{aligned} \frac{4u_{ij} - u_{i+1,j} - u_{i-1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} &= f_{ij} \quad \text{for} \quad i, j = 1, \ldots, n-1, \\ u_{0j} = u_{nj} = u_{i0} = u_{in} &= 0, \end{aligned} \tag{2.36}$$

where u_{ij} is now the approximate solution to the PDE at (x_i, y_j). As illustrated by Figure 2.7 each unknown variable is coupled only to its direct neighbors. It is clear from the discretization that there are $(n-1)^2$ degrees of freedom associated with the $(n-1)^2$ interior grid points. Using standard lexicographical ordering, the unknowns at the grid points in Ω_h may be collected into the vector

$$\mathbf{u} = (u_{11}, \ldots, u_{1,n-1}, \ldots, u_{n-1,1}, \ldots, u_{n-1,n-1})^\top.$$

Proceeding similarly for the source term produces the vector

$$\mathbf{f} = (f_{11}, \ldots, f_{1,n-1}, \ldots, f_{n-1,1}, \ldots, f_{n-1,n-1})^\top.$$

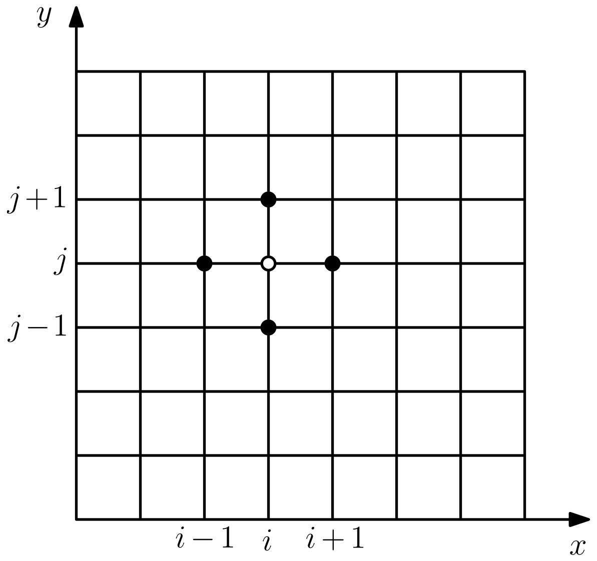


Figure 2.7: Two-dimensional grid on the unit square. The solid dots indicate the unknowns that are related at an interior grid point (x_i, y_j) by the discrete equation (2.36).

Thus the discrete problem can be formulated as an $(n-1)^2 \times (n-1)^2$ sparse linear system

$$\mathbf{A}\mathbf{u} = \mathbf{f} \tag{2.37}$$

in which $\mathbf{A}$ is the block tridiagonal matrix given by

$$\mathbf{A} = \frac{1}{h^2}\begin{bmatrix} \mathbf{T} & -\mathbf{I} & & \\ -\mathbf{I} & \mathbf{T} & \ddots & \\ & \ddots & \ddots & -\mathbf{I} \\ & & -\mathbf{I} & \mathbf{T} \end{bmatrix} \quad \text{with} \quad \mathbf{T} = \begin{bmatrix} 4 & -1 & & \\ -1 & 4 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{bmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}.$$

The matrix $\mathbf{A}$ is clearly symmetric, and in fact $\mathbf{A}$ is positive definite because

$$h^2\mathbf{x}^\top\mathbf{A}\mathbf{x} = \sum_i \sum_{j<i} |a_{ij}|(x_i - x_j)^2 + \sum_i {x_i}^2\Big(4 - \sum_{j\neq i} |a_{ij}|\Big) > 0 \ \text{ for all } \ \mathbf{x} \in \mathbb{R}^{(n-1)^2} \setminus \{\mathbf{0}\}.$$

Moreover, by Theorem 2.2.7 (4) $\mathbf{A}$ is a nonsingular M-matrix. As a consequence, the (weighted) Jacobi and Gauss–Seidel methods converge when applied to (2.37).

Now suppose we apply weighted Jacobi with $\omega = 2/3$ and Gauss–Seidel to equation (2.37) with $\mathbf{f} = \mathbf{0}$. (We consider weighted Jacobi and Gauss–Seidel because they are common choices for the *relaxation method* in multigrid algorithms.) We note that when examining the error of stationary iterative methods applied to nonsingular linear systems it is sufficient to work with the homogeneous system $\mathbf{Au} = \mathbf{0}$ and an arbitrary initial guess. A benefit of working with the homogeneous system is that the error of an approximate solution $\mathbf{v}$ is simply $-\mathbf{v}$. Figure 2.8 illustrates the error smoothing properties of weighted Jacobi and Gauss–Seidel. It is evident that while the error does not necessarily become small after a few iterations, it certainly becomes smooth. In the context of multigrid, the naming convention of "smoother" or "relaxation method" is in reference to the smoothing property of simple stationary iterative methods.

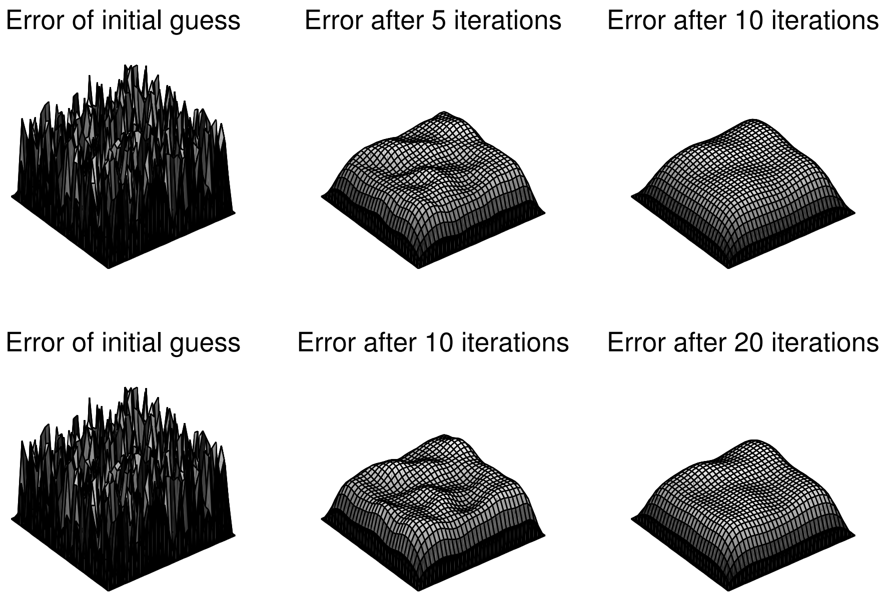


Figure 2.8: Smoothing properties of Gauss–Seidel and weighted Jacobi ($\omega = 2/3$) applied to $\mathbf{Au} = \mathbf{0}$ with the same random initial guess. The top row is Gauss–Seidel and the bottom row is weighted Jacobi.

Further insight into the smoothing property of simple stationary iterative methods is gained by considering the eigenmodes of the discrete Poisson problem (2.36) at the grid points in Ω_h:

$$\varphi_{ij}^{k,\ell} = \sin\left(\frac{k\pi i}{n}\right)\sin\left(\frac{\ell\pi j}{n}\right) \quad \text{for} \quad i,j = 1,\dots,n-1 \tag{2.38}$$

and $k,\ell \in \{1,\dots,n-1\}$. Collecting these values into the vector $\boldsymbol{\varphi}^{k,\ell}$ for each wavenumber (k,ℓ), it is clear that $\boldsymbol{\varphi}^{1,1},\dots,\boldsymbol{\varphi}^{n-1,n-1}$ are the eigenvectors of $\mathbf{A}$, which form a basis for $\mathbb{R}^{(n-1)^2}$. This observation motivates us to consider the smoothing properties of weighted Jacobi and Gauss–Seidel when applied to the homogeneous system $\mathbf{Au} = \mathbf{0}$ with $\boldsymbol{\varphi}^{k,\ell}$ as the initial guess (the negative of initial error). Figure 2.9 gives the number of iterations required by weighted Jacobi ($\omega = 2/3$) and Gauss–Seidel to reduce the error by a factor of 10^3. Figure 2.9 clearly illustrates that for the discrete Poisson problem (2.36) stationary

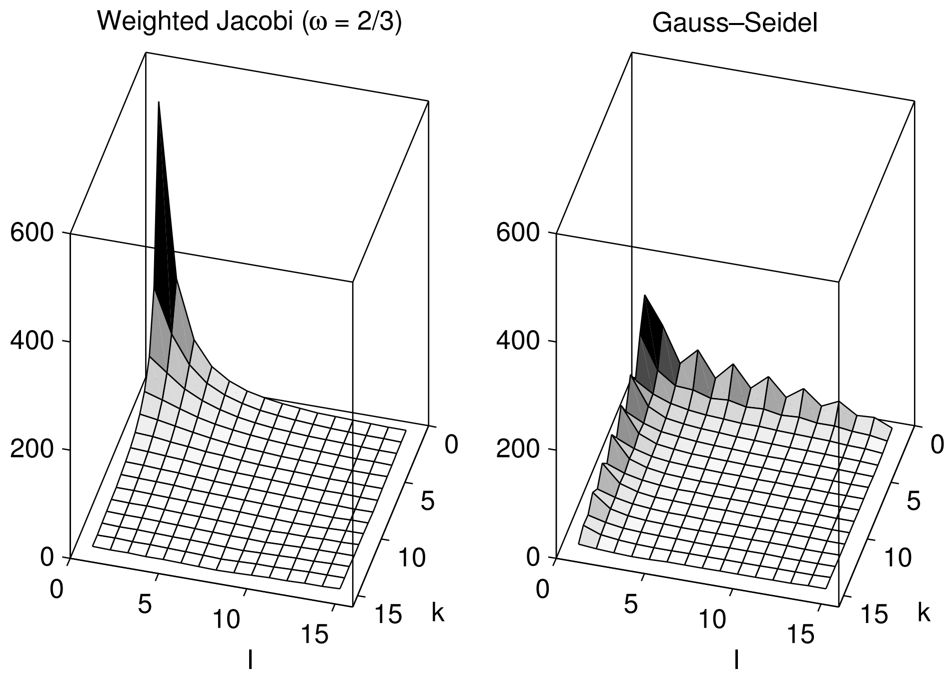


Figure 2.9: Number of iterations needed to reduce the initial error $\boldsymbol{\varphi}^{k,\ell}$ by a factor of 10^3 on a 15×15 grid.

iterative methods are more effective at reducing the high frequency or oscillatory error components (large k or ℓ) than the low frequency or smooth error components (small k and ℓ). Note that the relaxation parameter $\omega = 2/3$ is near the optimal value of $4/5$ for reducing high frequency error components for this problem (see [119]).

We have seen that the weighted Jacobi and Gauss–Seidel methods possess a smoothing property, and are particularly effective at removing oscillatory components of the error. This observation raises the question of whether these methods can also be used to attenuate smooth components of the error. In multigrid this is accomplished by relaxing on a *coarse grid*. Since there are fewer degrees of freedom on a coarse grid, relaxations on a coarse grid require less work. Suppose we have relaxed on the *fine grid* Ω_h and are confident that only smooth error remains. Then the question we must ask ourselves is if the smooth error (low frequency eigenmodes) can be accurately represented on a coarse grid. The answer to this question is the second principle of multigrid, namely the *coarse grid principle*. Assuming that n is an even number, a natural way of defining the coarse grid is to double the grid spacing in each dimension, which is referred to as *standard coarsening*.The coarse grid obtained by standard coarsening is given by

$$\Omega_{2h} = \{(x_i, y_j) : x_i = 2hi,\ y_j = 2hj,\ 1 \leq i, j \leq n/2 - 1\}.$$

Now consider the eigenmodes of the discrete Poisson problem (2.38). We observe that

$$\varphi_{ij}^{k,\ell} = -\varphi_{ij}^{n-k,\ell} = -\varphi_{ij}^{k,n-\ell} = \varphi_{ij}^{n-k,n-\ell} \quad \text{for} \quad i, j = 2, 4, 6, \ldots, n-2. \tag{2.39}$$

Hence, on the coarse grid Ω_{2h} any eigenmode with $n/2 < \max\{k, \ell\} < n$ coincides with an eigenmode for which $0 < \max\{k, \ell\} < n/2$. Therefore, the high frequency modes are not "visible" on the coarse grid, a phenomenon referred to as *aliasing*. We note that for $k = n/2$ or $\ell = n/2$, the $\varphi_{ij}^{k,\ell}$ vanish on Ω_{2h}. Therefore, it is exactly the low frequency eigenmodes that can be represented on the coarse grid Ω_{2h}. Furthermore, because there is only one quarter the number of points on the coarse grid as on the fine grid, the low frequency eigenmodes on Ω_h must appear more oscillatory on Ω_{2h} (see Figure 2.10). Consequently, relaxation on the coarse grid should be more effective at removing smooth error. We note that the definition of “high frequency” and “low frequency” modes depends on the fine grid as well as on the coarse grid. For example, suppose the grid spacing on the coarse grid is $4h$ (assuming n is divisible by 4) instead of $2h$. Then by a similar argument as above with $i, j = 4, 8, 12, \ldots, n-4$, the low frequency modes correspond to $0 < \max\{k, \ell\} < n/4$, and the high frequency modes correspond to $n/4 < \max\{k, \ell\} < n$.

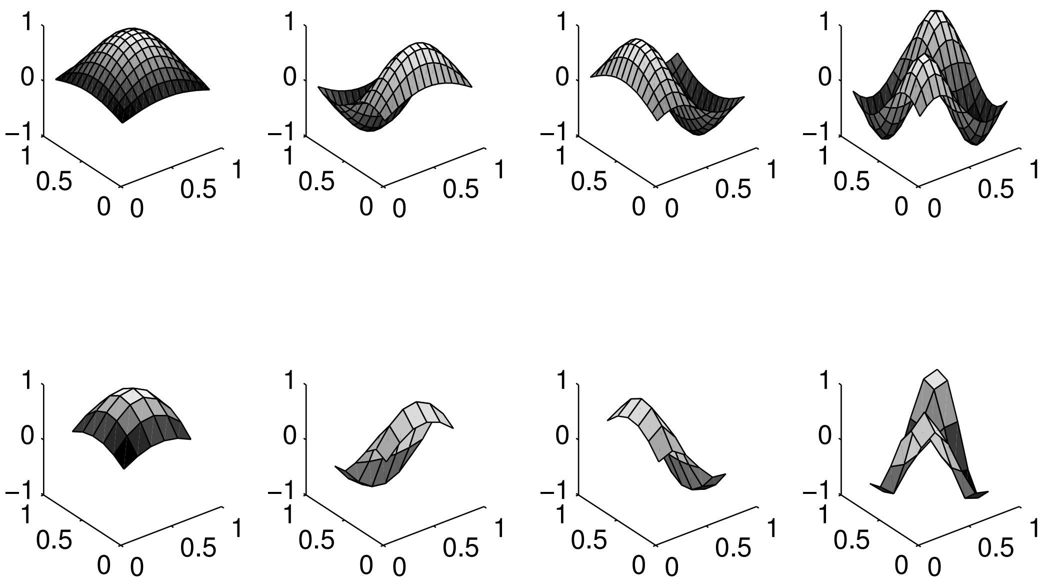


Figure 2.10: Low frequency eigenmodes $\boldsymbol{\varphi}^{1,1}, \boldsymbol{\varphi}^{1,2}, \boldsymbol{\varphi}^{2,1}, \boldsymbol{\varphi}^{2,2}$ on a 16×16 grid (top, right to left). The same low frequency eigenmodes on a coarser 8×8 grid (bottom, right to left).

2.6.2 The multigrid algorithm

Building on the discussion in the previous section, we describe the individual components of a multigrid method, in the context of our model problem, and show how they may be combined into a single algorithmic unit. Consider the linear system

$$\mathbf{A}_h \mathbf{u}_h = \mathbf{f}_h \tag{2.40}$$

corresponding to the discrete Poisson equation (2.36) defined on the grid Ω_h. Let Ω_{2h} denote the coarse version of the fine grid Ω_h obtained by standard coarsening. There are $N_h = (n-1)^2$ degrees of freedom on Ω_h, and $N_{2h} = (n/2-1)^2$ degrees of freedom on Ω_{2h}. Now recall the two basic principles of multigrid that were motivated in Section 2.6.1.

Smoothing Principle: Simple stationary iterative methods such as weighted Jacobi and Gauss–Seidel, when applied to linear systems with suitable structure, have a strong smoothing effect on the error of any approximation.

Coarse Grid Principle: A smooth error term can be well approximated on a coarse grid. Because the coarse grid has fewer degrees of freedom, it is much cheaper to do work on the coarse grid.

Suppose we have an initial approximation $\tilde{\mathbf{u}}_h$ to $\mathbf{u}_h$ such that its corresponding error $\mathbf{e}_h = \mathbf{u}_h - \tilde{\mathbf{u}}_h$ is sufficiently smooth. Then by the coarse grid principle the error can be accurately approximated on the coarse grid, where relaxations are cheaper to perform and are more effective at removing smooth error modes. Moreover, provided the error is accurately approximated on the coarse grid and there is some way of transferring it to the fine grid, adding the fine-grid error approximation to $\tilde{\mathbf{u}}_h$ should yield an improved solution $\tilde{\mathbf{u}}_h + \tilde{\mathbf{e}}_h \approx \mathbf{u}_h$. The procedure we have just loosely described is referred to as a *coarse-grid correction scheme*, and it has the makings of a multigrid method. However, some questions now arise. First and foremost, how do we relax the error on Ω_{2h}? On the fine grid, the error is related to the residual $\mathbf{r}_h = \mathbf{f}_h - \mathbf{A}_h\tilde{\mathbf{u}}_h$ by the residual equation

$$\mathbf{A}_h\mathbf{e}_h = \mathbf{r}_h. \tag{2.41}$$

Assuming the error is smooth it can be approximated on the coarse grid as the solution of a coarse residual equation

$$\mathbf{A}_{2h}\mathbf{e}_{2h} = \mathbf{r}_{2h}. \tag{2.42}$$

The error can then be relaxed on the coarse grid by applying the relaxation scheme to the coarse system (2.42). Because relaxing on the fine system (2.40) with an arbitrary initial guess $\tilde{\mathbf{u}}_h$ is equivalent to relaxing on the residual equation (2.41) with a zero initial guess, it follows that we may use $\mathbf{0}_{2h}$ as the initial guess for the error on the coarse grid.

To obtain the coarse residual $\mathbf{r}_{2h}$ we need some way of transferring fine grid quantities to the coarse grid. This is accomplished by defining a (linear) transfer operator

$$\mathbf{I}_h^{2h} : \mathbb{R}^{N_h} \to \mathbb{R}^{N_{2h}}, \tag{2.43}$$

referred to as the *restriction operator*. The coarse residual is then defined by

$$\mathbf{r}_{2h} \coloneqq \mathbf{I}_h^{2h}\mathbf{r}_h.$$

Similarly, we can define an *interpolation* or *prolongation* operator

$$\mathbf{I}_{2h}^{h} : \mathbb{R}^{N_{2h}} \to \mathbb{R}^{N_h}, \tag{2.44}$$

which transfers coarse grid quantities to the fine grid. In particular, the prolongation operator may be used to obtain a fine grid approximation of the coarse grid error

$$\tilde{\mathbf{e}}_h = \mathbf{I}_{2h}^{h}\mathbf{e}_{2h}.$$

The most basic restriction operator is the *injection* operator, which simply assigns fine-grid values located at the coarse-grid points to the coarse grid. For example, using injection to define the coarse residual for the model problem we have that

$$(r_{2h})_{ij} = (r_h)_{2i,2j} \quad \text{for} \quad i,j = 1,\dots,\frac{n}{2}-1.$$

While there are many different choices for the restriction and interpolation operators, usually the simplest of these are effective. Common transfer operators for the model problem with standard coarsening are *bilinear* interpolation, and *full weighting* restriction, which corresponds to the scaled transpose of the bilinear interpolation operator. Figure 2.11 illustrates the distribution process for the bilinear interpolation operator. In general, prolongation corresponds to *piecewise multilinear interpolation*, which is the extension of

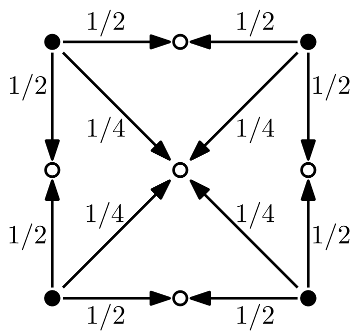


Figure 2.11: Distribution process for the bilinear interpolation operator on a uniform grid. Solid black dots (●) are coarse grid points and hollow dots (○) are fine grid points.

piecewise linear interpolation to higher dimensions. In the case of bilinear interpolation, interpolated values are given by averages of the values at neighboring grid points. Owing to the simple structure of these types of prolongation, *smooth* fine-grid errors $\mathbf{e}_h$ can be well approximated by coarse-grid interpolation, whereas oscillatory fine-grid errors cannot. Because the precise structure of the transfer operators is not important to us here, we refer to [119] for further details.

Because the coarse grid has the same structure as the fine grid, only with fewer grid points, the *coarse-grid system operator* $\mathbf{A}_{2h}$ can be obtained by discretizing the problem on Ω_{2h}. An alternative approach is to use the *Galerkin* formulation of the coarse-grid system operator, given by

$$\mathbf{A}_{2h} = \mathbf{I}_h^{2h} \mathbf{A}_h \, \mathbf{I}_{2h}^h.$$

It can be shown that if $\mathbf{A}_h$ is symmetric positive definite, then the Galerkin operator with $\mathbf{I}_h^{2h} = (\mathbf{I}_{2h}^h)^\top$ satisfies an important variational property involving the coarse-grid correction (see Theorem 2.6.1).

We now have all the ingredients to precisely describe a *two-grid correction scheme*, which is a combination of the coarse-grid correction procedure described above, together with pre- and post-smoothing steps (Algorithm 2.1). It is the complementary relationship of these two processes that makes multigrid methods so effective.

Algorithm 2.1: Two-grid correction scheme for $\mathbf{A}_h\mathbf{u}_h = \mathbf{f}_h$

```
    Input: current approximation $\mathbf{u}_h^{(k)}$, $\mathbf{A}_h$, $\mathbf{f}_h$, number of smoothing steps $\nu_1$, $\nu_2$
    Output: New approximation $\mathbf{u}_h^{(k+1)}$
1.  Pre-smoothing: apply $\nu_1$ relaxations to $\mathbf{u}_h^{(k)}$
        $\bar{\mathbf{u}}_h \leftarrow \text{Relax}(\mathbf{A}_h, \mathbf{f}_h, \mathbf{u}_h^{(k)}, \nu_1)$
2.  Coarse-grid correction:
3.      Compute coarse-grid residual $\mathbf{r}_{2h} \leftarrow \mathbf{I}_h^{2h}(\mathbf{f}_h - \mathbf{A}_h\bar{\mathbf{u}}_h)$
4.      Solve $\mathbf{A}_{2h}\mathbf{e}_{2h} = \mathbf{r}_{2h}$
5.      Interpolate the coarse-grid error to the fine grid $\hat{\mathbf{e}}_h \leftarrow \mathbf{I}_{2h}^h\mathbf{e}_{2h}$ and
        correct the fine grid approximation $\mathbf{u}_h^{\text{CGC}} \leftarrow \bar{\mathbf{u}}_h + \hat{\mathbf{e}}_h$
6.  Post-smoothing: apply $\nu_2$ relaxations to $\mathbf{u}_h^{\text{CGC}}$
        $\mathbf{u}_h^{(k+1)} \leftarrow \text{Relax}(\mathbf{A}_h, \mathbf{f}_h, \mathbf{u}_h^{\text{CGC}}, \nu_2)$
```

Several comments regarding Algorithm 2.1 are in order. First, we use

$$\text{Relax}(\mathbf{A}, \mathbf{f}, \mathbf{x}^{(0)}, \nu)$$

to denote ν iterations of a stationary iterative procedure applied to the linear system $\mathbf{Ax} = \mathbf{f}$ with initial guess $\mathbf{x}^{(0)}$. Common choices for the relaxation scheme include the weighted Jacobi and Gauss–Seidel methods. The integers ν_1 and ν_2, which control the number of pre- and post-smoothing steps, are typically fixed prior to the start of the method. In most cases only a few pre- and post-smoothing steps are necessary on each grid level, for example, $\nu_1, \nu_2 \in \{1, 2, 3\}$.

It is important to appreciate the complementarity inherent to the two-grid correction scheme. Pre-smoothing reduces the high frequency components of the error, leaving the error smooth. Thus, the error after smoothing can be accurately approximated on a coarse grid. Assuming the coarse residual equation is solved accurately, the interpolated error should be a good approximation of the fine-grid error, and correction of the fine-grid solution should be effective. Post-smoothing helps reduce any high frequency error components introduced by interpolation.

The two-grid method serves as a basis for a true multigrid method, and is particularly important as a theoretical tool for proving (two-grid) convergence (see [119]). The multigrid idea arises from the observation that since the coarse-grid equation (2.42) may be too large and hence too costly to solve directly, it is more effective and efficient to obtain an approximate solution of (2.42) by recursively applying the two-grid method with a sequence of coarser grids. This approach is possible because the coarse-grid equation is of the same form as the original equation. Consider the $n \times n$ linear system

$$\mathbf{Au} = \mathbf{f}$$

arising from the discretization of a boundary value problem on the grid Ω_0, with grid spacing h, and suppose we define a sequence of consecutively coarser grids

$$\Omega_0, \Omega_1, \ldots, \Omega_\ell, \ldots, \Omega_L,$$

with Ω_L as the coarsest grid. Moreover, let n_ℓ be the number of unknowns on Ω_ℓ for $\ell = 0, \ldots, L$ with $n_0 = n$. For example, in the case of the model problem with standard coarsening, Ω_ℓ would correspond to the uniform grid with mesh size $h_\ell = 2^{-\ell}h$. Suppose that for each grid Ω_ℓ the linear operators

$$\mathbf{I}_\ell^{\ell+1} : \mathbb{R}^{n_\ell} \longrightarrow \mathbb{R}^{n_{\ell+1}}, \quad \mathbf{I}_{\ell+1}^{\ell} : \mathbb{R}^{n_{\ell+1}} \longrightarrow \mathbb{R}^{n_\ell}, \quad \mathbf{A}_\ell \in \mathbb{R}^{n_\ell \times n_\ell}$$

are given, where $\mathbf{A}_\ell \mathbf{u}_\ell = \mathbf{f}_\ell$ is the discrete problem to be solved on Ω_ℓ, and $\mathbf{A}_0 = \mathbf{A}$. A *multigrid cycle* for $\mathbf{Au} = \mathbf{f}$ is given by Algorithm 2.2. The input parameter μ controls the number of recursive solves performed on each level, and is referred to as the *cycle index*. In practice, only $\mu = 1$ or 2 are used, where the multigrid cycle is referred to as a *V-cycle* for $\mu = 1$, and a *W-cycle* for $\mu = 2$. While W-cycles are computationally more expensive than V-cycles, they typically result in faster convergence than V-cycles, so the goal is to choose the cycle that leads to the fastest overall execution. We note that other cycling schemes are possible, however, V-cycles and W-cycles are the most common in practice. The structure of a multigrid V-cycle and W-cycle is illustrated by Figure 2.12. A multigrid method in its most basic incarnation consists of a sequence of V-cycles or W-cycles, where the output of the kth cycle (iteration) is used as input for the $(k+1)$th cycle. Algorithm 2.2 recursively visits successively coarser grids until it reaches the coarsest grid Ω_L where the corresponding set of equations are solved directly. The problem on the coarsest level is usually invertible and hence amenable to solution by LU factorization, however, its exact nature depends on the underlying continuous problem and on the discretization. The size of Ω_L should be sufficiently small so that the direct solve is computationally inexpensive. We note that in lieu of computing the exact solution on Ω_L, it may be sufficient to obtain an approximate solution by applying a number of relaxations on Ω_L.

Algorithm 2.2: MGCYC (generic multigrid cycle for solving $\mathbf{Au} = \mathbf{f}$)

```
    Input: $\mathbf{u}_\ell^{(k)}$, $\mathbf{A}_\ell$, $\mathbf{f}_\ell$, $\mu$, $\nu_1$, $\nu_2$
    Output: $\mathbf{u}_\ell^{(k+1)}$
    if $\ell = L$ then
1.      Solve $\mathbf{A}_L\mathbf{u}_L = \mathbf{f}_L$
    else
2.      Pre-smoothing: apply $\nu_1$ relaxations to $\mathbf{u}_\ell^{(k)}$
            $\bar{\mathbf{u}}_\ell \leftarrow \mathrm{Relax}(\mathbf{A}_\ell, \mathbf{f}_\ell, \mathbf{u}_\ell^{(k)}, \nu_1)$
3.      Restrict the residual: $\mathbf{f}_{\ell+1} \leftarrow \mathbf{I}_\ell^{\ell+1}(\mathbf{f}_\ell - \mathbf{A}_\ell\bar{\mathbf{u}}_\ell)$
4.      Set $\mathbf{v}_{\ell+1} \leftarrow \mathbf{0}$ and repeat $\mu \geq 1$ times:
            $\mathbf{v}_{\ell+1} \leftarrow \mathrm{MGCYC}(\mathbf{v}_{\ell+1}, \mathbf{A}_{\ell+1}, \mathbf{f}_{\ell+1}, \mu, \nu_1, \nu_2)$
5.      Correct: $\mathbf{u}_\ell^{\mathrm{CGC}} \leftarrow \bar{\mathbf{u}}_\ell + \mathbf{I}_{\ell+1}^{\ell}\mathbf{v}_{\ell+1}$
6.      Post-smoothing: apply $\nu_2$ relaxations to $\mathbf{u}_\ell^{\mathrm{CGC}}$
            $\mathbf{u}_\ell^{(k+1)} \leftarrow \mathrm{Relax}(\mathbf{A}_\ell, \mathbf{f}_\ell, \mathbf{u}_\ell^{\mathrm{CGC}}, \nu_2)$
    end
```

We observe from Algorithm 2.2 that a multigrid cycle comprises:

1. A sequence of consecutively coarser grids $\Omega_0, \ldots, \Omega_L$
2. A set of inter-grid transfer operators defined on $\Omega_0, \ldots, \Omega_{L-1}$
3. A set of system operators $\mathbf{A}_\ell$ defined on Ω_ℓ for $\ell = 0, \ldots, L$
4. A relaxation method
5. A coarsest-grid solver

Together, these components form what is called a *multigrid hierarchy.*

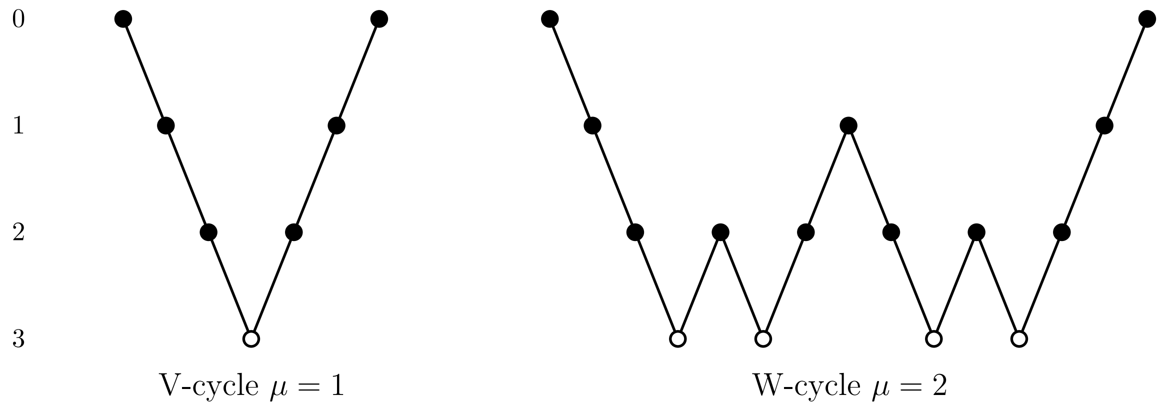


Figure 2.12: Structure of a multigrid V-cycle and W-cycle with $L = 3$ ($\bullet$ smoothing, $\circ$ coarsest level solve, $\backslash$ coarse-to-fine transfer, / fine-to-coarse transfer).

Remark 2.6.1. Multigrid methods can be applied most efficiently to *elliptic* boundary value problems, of which the Poisson problem is a prototypical example. The multigrid approach may also be applied to boundary value problems involving *parabolic* and *hyperbolic* PDEs, however, doing so may require special considerations that are beyond the scope of this thesis. In particular, care must be taken to choose an appropriate discretization of the differential operator as the discretization directly influences the properties of the corresponding system of equations $\mathbf{Au} = \mathbf{f}$. For simplicity, in this section we have assumed that $\mathbf{Au} = \mathbf{f}$ arises from the discretization of a linear elliptic boundary value problem and that the discrete operator $\mathbf{A}$ is a symmetric positive definite matrix.

We conclude this section by analyzing the computational work of a multigrid cycle. Following [119], the computational work W_0 per multigrid cycle is recursively given by

$$W_{L-1} = W_{L-1}^{L} + W_L, \quad W_{\ell-1} = W_{\ell-1}^{\ell} + \mu W_\ell \text{ for } \ell = L-1, \ldots, 1 \tag{2.45}$$

where $W_{\ell-1}^{\ell}$ denotes the computational work of a two-grid cycle with fine grid $\Omega_{\ell-1}$ and coarse grid Ω_ℓ, excluding the work needed to solve the residual equation on Ω_ℓ, and W_L is the amount of work to compute the exact solution on Ω_L. It follows by (2.45) that

$$W_0 = \mu^{L-1} W_L + \sum_{\ell=1}^{L-1} \mu^{\ell-1} W_{\ell-1}^{\ell} \tag{2.46}$$

for $L \geq 1$. We assume that $W_{\ell-1}^{\ell}$ is proportional to the number of points on $\Omega_{\ell-1}$, that is, $n_{\ell-1}$, where the constant of proportionality C is given by

$$C \approx (\nu_1 + \nu_2) w_s + w_c + w_r.$$

Here, w_s, w_c and w_r are estimates of the amount of work per grid point of $\Omega_{\ell-1}$ to perform a single smoothing step, compute the correction, and compute the coarse residual, respectively. Note that "$\approx$" means "$=$" up to lower-order terms. We further assume that the coarsening is such that $n_{\ell-1} \approx \alpha n_\ell$ for $\ell = 1, \ldots, L$ with $\alpha > 1$. For example, in the case of standard coarsening in two dimensions $\alpha = 4$. Putting all the pieces together it follows that

$$W_0 \approx \sum_{\ell=1}^{L-1} Cn \left(\frac{\mu}{\alpha}\right)^{\ell-1} + \mu^{L-1} W_L < \frac{\alpha}{\alpha-\mu} Cn + \mu^{L-1} W_L \text{ for } \mu < \alpha. \tag{2.47}$$

Assuming that the amount of work to compute the exact solution on Ω_L is negligible, we conclude that

$$W_0 \lessapprox \frac{\alpha}{\alpha-\mu} Cn \text{ for } \mu < \alpha \tag{2.48}$$

where "$\lessapprox$" means "$\leq$" up to lower order terms. Therefore, in the case of standard coarsening in two dimensions, it follows that the computational work of a V-cycle or a W-cycle is proportional to the number of points on the finest grid.

Remark 2.6.2. In many cases it can be shown through *local Fourier analysis* that the convergence speed of an *appropriate* multigrid method does not depend on the size of the finest grid. For example, see chapters 3 and 4 in [119] as well as [18]. By "appropriate" we mean a multigrid method in which suitable care has been taken in choosing each of its components: transfer operators, smoother, coarsening. Together with the fact that each multigrid cycle requires only $\mathcal{O}(n)$ arithmetic operations, it follows that multigrid methods are capable of achieving a fixed reduction of the error in $\mathcal{O}(n)$ arithmetic operations. This is the real "magic" of multigrid, often referred to as the "optimality" of multigrid methods.

In this section and the previous one we described the basic principles of multigrid and presented a generic multigrid cycle. Unfortunately, designing a multigrid method is not so simple as throwing together a smoother, a coarsening strategy and some transfer operators. Care must be taken in selecting these components so they work in unison to produce an efficient and robust algorithm, and at the same time account for the various subtleties of the underlying problem. In particular, because the grid hierarchy is often fully determined by a specific coarsening strategy, special care must be taken in choosing the relaxation method so that an efficient interplay between smoothing and coarse-grid correction is obtained. Thankfully, theoretical tools such as *smoothing analysis* and *local Fourier analysis* exist to help guide the selection of these various components [119]. In the next section we describe *algebraic multigrid*, a variation of the "geometric" multigrid method considered here, which relies on the same basic multigrid principles, but is intended for problems in which the underlying grid is unstructured, or there is no underlying grid.

2.6.3 Classical algebraic multigrid (AMG)

In contrast to *geometric multigrid*, in which the underlying differential equation and geometry of the problem at hand are used to guide the solution process of the discretized equations $\mathbf{Au} = \mathbf{f}$, *algebraic multigrid* (AMG) bases multigrid concepts solely on information contained in the coefficient matrix of the algebraic system. In particular, coarse grids, coarse-grid system operators, and inter-grid transfer operators (i.e., interpolation and restriction operators) are *automatically* constructed, based only on the elements in the current fine-grid system operator $\mathbf{A}_h$. Consequently, AMG can be used as a solver for problems in which no structured grids are employed, for example, finite-element discretization with irregular fine-grid triangulations, and large sparse systems that are not derived from differential equation problems. AMG may also useful for boundary value problems in which conventional coarsening may not be able to account for certain pathologies in the equation coefficients, for example, diffusion problems with discontinuous coefficients.

The main conceptual difference between geometric multigrid and AMG lies in the interplay between the coarse-grid correction process and the smoothing process. Because geometric multigrid employs a prescribed grid hierarchy, an appropriate smoothing process must be chosen to ensure an efficient interplay with the coarse-grid correction. In contrast, as discussed in [119], AMG fixes the smoother to a simple relaxation scheme, and enforces an efficient interplay with the coarse-grid correction by choosing the coarse grids and interpolation operators appropriately.

The first "AMG-like" method, which employed *operator-dependent interpolation* (interpolation based on the current fine-grid system operator) in an otherwise geometric multigrid setting, was introduced in 1981 by Alcouffe, Brandt, Dendy, and Painter [2] to solve diffusion problems with strongly discontinuous coefficients. Classical algebraic multigrid (as described below) was developed in the early-to-mid 1980's by Brandt, McCormick, Ruge, and Stüben [19, 22, 24, 103]. Since algebraic multigrid's inception, a large amount of research has been focused on advancing AMG theory and on the development of new multigrid algorithms that extend the applicability of AMG to new classes of problems, for example, see [21, 26, 28, 29, 30, 31, 66, 96, 121, 124]. For a brief introduction to AMG that outlines many of the main developments since the initial papers on AMG we recommend [56]. For a more thorough introduction to AMG, see [32, 119].

In the remainder of this section we describe the classical AMG algorithm developed by Brandt, McCormick, Ruge, and Stüben for solving the $n \times n$ sparse linear system of equations

$$\mathbf{A}\mathbf{u} = \mathbf{f},$$

where it is assumed for simplicity that $\mathbf{A}$ is a symmetric nonsingular M-matrix with nonnegative row sums[3]. We note that AMG is applicable to other classes of matrices including *essentially positive-type* matrices in which some of the off-diagonal elements can be positive, providing they are sufficiently small (see [19]).

Analogous to geometric multigrid, in AMG the index of each variable u_i is associated with a "grid point" that belongs to the "grid" $\Omega = \{1, \dots, n\}$. Connections between grid points in Ω may then be defined as the edges in the undirected adjacency graph of $\mathbf{A}$, denoted by $\mathcal{G}(\mathbf{A}) = \mathcal{G}(\Omega, \mathcal{E})$, where an edge exists between node i and node j if $i \neq j$ and $a_{ij} \neq 0$ (see Figure 2.13). Points on the "coarse grid" are given by a subset of the

$$\mathbf{A} = \begin{bmatrix} * & * & * & & * \\ * & * & * & & \\ * & * & * & * & \\ & & * & * & * \\ * & & & * & * \end{bmatrix}$$

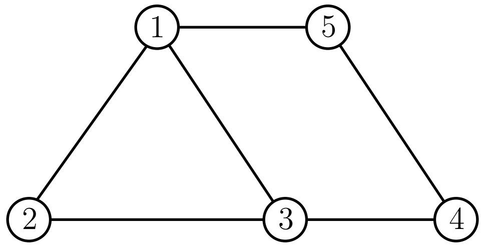


Figure 2.13: Undirected graph $\mathcal{G}(\mathbf{A})$ of the matrix $\mathbf{A}$. Nonzero elements in $\mathbf{A}$ are indicated by asterisks.

fine-grid points $\mathcal{C} \subset \Omega$, and hence a subset of fine-grid variables $\{u_i : i \in \mathcal{C}\}$ serve as the coarse-grid unknowns. While this artificial geometry has no physical meaning, it provides a means of defining connections between points as well as local neighborhoods of points, in an algebraic setting.

Definition 2.6.1. For any point i, the (direct) *neighborhood* of i is defined as the set of points that are adjacent to i in $\mathcal{G}(\mathbf{A})$, that is, $\mathcal{N}_i = \{j \neq i : a_{ij} \neq 0\}$.

[3]This assumption is equivalent to the requirement that $\mathbf{A}$ is a weakly row diagonally dominant, symmetric, positive definite matrix with nonpositive off-diagonal elements.

The *locality* of two points i and j corresponds to the distance between nodes i and j in $\mathcal{G}(\mathbf{A})$, defined as number of edges in the shortest path connecting them. For instance, $\mathcal{N}_i$ consists of the distance one connections to node i.

Similar to geometric multigrid, an AMG hierarchy consists of the following components:

1. A sequence of consecutively coarser grids $\Omega_0 \supset \Omega_2 \supset \cdots \supset \Omega_L$ with $\Omega_0 = \Omega$
2. A set of transfer operators
   a. Interpolation: $\mathbf{I}_{\ell+1}^{\ell} : \mathbb{R}^{n_{\ell+1}} \longrightarrow \mathbb{R}^{n_\ell}$ for $\ell = 0, \ldots, L-1$
   b. Restriction: $\mathbf{I}_{\ell}^{\ell+1} : \mathbb{R}^{n_\ell} \longrightarrow \mathbb{R}^{n_{\ell+1}}$ for $\ell = 0, \ldots, L-1$
3. A set of system operators $\mathbf{A}_\ell \in \mathbb{R}^{n_\ell \times n_\ell}$ for $\ell = 0, \ldots, L$ with $\mathbf{A}_0 = \mathbf{A}$
4. A smoother such as weighted Jacobi or Gauss–Seidel
5. A coarsest-grid solver

Remark 2.6.3. When considering two consecutive levels only, we use indices h and H instead of ℓ and $\ell + 1$, respectively, to distinguish between fine-grid and coarse-grid quantities.

Once the components of the AMG hierarchy are defined, the recursively defined AMG cycle is given by Algorithm 2.2. For this cycle to work efficiently, relaxation and coarse-grid correction must work together to effectively reduce all error components. As discussed above, AMG fixes the relaxation scheme and appropriately chooses the coarse grids, the transfer operators, and the system operators to ensure an effective interplay with the coarse-grid correction. The following principles guide the choice of these components:

P1: Error components not efficiently removed by relaxation must be well approximated by the range of interpolation.

P2: The coarse-grid problem must provide a good approximation to the fine-grid error in the range of interpolation.

Principle **P1** determines the coarse grids and interpolation. To satisfy **P1**, the coarse grid and interpolation are to be selected automatically so that the range of interpolation accurately approximates error components that are slow to converge by relaxation. Principle **P2** affects the choice of restriction and the coarse-grid system operator. For symmetric positive definite operators, **P2** is satisfied by using the *Galerkin* formulation for the restriction and coarse-grid system operators:

$$\mathbf{A}_{\ell+1} = \mathbf{I}_{\ell}^{\ell+1} \mathbf{A}_{\ell} \, \mathbf{I}_{\ell+1}^{\ell} \quad \text{and} \quad \mathbf{I}_{\ell}^{\ell+1} = (\mathbf{I}_{\ell+1}^{\ell})^{\top}. \tag{2.49}$$

When $\mathbf{A}$ is symmetric positive definite, Galerkin-based coarse-grid corrections satisfy a variational principle in that they minimize the energy norm of the error with respect to all error components in the range of interpolation. That is, the correction from the exact solution of the coarse-grid problem is the best approximation (with respect to the energy norm) in the range of interpolation. A further useful property of (2.49) is that $\mathbf{A}_{\ell+1}$ is a symmetric positive definite operator if the interpolation operator is of full rank. To demonstrate this variational principle, consider a (two-grid) coarse-grid correction process between consecutive fine and coarse grids $\Omega_h \supset \Omega_H$,

$$\begin{aligned}
\mathbf{u}_h^{\text{CGC}} &= \bar{\mathbf{u}}_h + \mathbf{I}_H^h \mathbf{v}_H \\
&= \bar{\mathbf{u}}_h + \mathbf{I}_H^h ((\mathbf{A}_H)^{-1} \mathbf{I}_h^H (\mathbf{f}_h - \mathbf{A}_h \bar{\mathbf{u}}_h)) \\
&= (\mathbf{I}_h - \mathbf{I}_H^h (\mathbf{A}_H)^{-1} \mathbf{I}_h^H \mathbf{A}_h) \bar{\mathbf{u}}_h + \mathbf{I}_H^h (\mathbf{A}_H)^{-1} \mathbf{I}_h^H \mathbf{f}_h.
\end{aligned}$$

Expressed in terms of the corresponding error we have that

$$\mathbf{e}_h^{\text{CGC}} = \mathbf{C}_{h,H} \, \bar{\mathbf{e}}_h \quad \text{with} \quad \mathbf{C}_{h,H} \coloneqq \mathbf{I}_h - \mathbf{I}_H^h (\mathbf{A}_H)^{-1} \mathbf{I}_h^H \mathbf{A}_h, \tag{2.50}$$

where $\mathbf{C}_{h,H}$ is the (two-grid) *coarse-grid correction operator*. Assuming that $\mathbf{A}_H$ is given by the Galerkin formulation in (2.49), we have the following proposition.

Proposition 2.6.1. *The Galerkin-based coarse-grid correction operator* $\mathbf{C}_{h,H}$ *is an* $\mathbf{A}_h$-*orthogonal projection such that* $\text{range}(\mathbf{C}_{h,H}) \perp_{\mathbf{A}_h} \text{range}(\mathbf{I}_H^h)$.

Proof. The coarse-grid correction operator is an $\mathbf{A}_h$-orthogonal projection, if (1) $\mathbf{C}_{h,H}$ is symmetric with respect to the energy inner product and (2) $(\mathbf{C}_{h,H})^2 = \mathbf{C}_{h,H}$. Condition (1) follows by the observation that

$$\mathbf{A}_h\mathbf{C}_{h,H} = \mathbf{A}_h - \mathbf{A}_h\mathbf{I}_H^h(\mathbf{A}_H)^{-1}\mathbf{I}_h^H\mathbf{A}_h = (\mathbf{I}_h - \mathbf{A}_h(\mathbf{I}_h^H)^\top(\mathbf{A}_H)^{-1}(\mathbf{I}_H^h)^\top)\mathbf{A}_h = (\mathbf{C}_{h,H})^\top\mathbf{A}_h.$$

Condition (2) follows by

$$\begin{aligned}(\mathbf{C}_{h,H})^2 &= \mathbf{I}_h - 2\mathbf{I}_H^h(\mathbf{A}_H)^{-1}\mathbf{I}_h^H\mathbf{A}_h + \mathbf{I}_H^h(\mathbf{A}_H)^{-1}\underbrace{\mathbf{I}_h^H\mathbf{A}_h\mathbf{I}_H^h}_{\mathbf{A}_H}(\mathbf{A}_H)^{-1}\mathbf{I}_h^H\mathbf{A}_h\\ &= \mathbf{I}_h - 2\mathbf{I}_H^h(\mathbf{A}_H)^{-1}\mathbf{I}_h^H\mathbf{A}_h + \mathbf{I}_H^h(\mathbf{A}_H)^{-1}\mathbf{I}_h^H\mathbf{A}_h\\ &= \mathbf{C}_{h,H}.\end{aligned}$$

Having verified that $\mathbf{C}_{h,H}$ is an $\mathbf{A}_h$-orthogonal projection, it must be true that

$$\operatorname{range}(\mathbf{C}_{h,H}) \perp_{\mathbf{A}_h} \operatorname{null}(\mathbf{C}_{h,H}) = \operatorname{range}(\mathbf{I}_h - \mathbf{C}_{h,H}).$$

Because $\mathbf{I}_H^h$, $\mathbf{A}_h$, and $\mathbf{A}_H$ are all of full rank, $(\mathbf{A}_H)^{-1}\mathbf{I}_h^H\mathbf{A}_h$ is also of full rank. Therefore,

$$\operatorname{range}(\mathbf{I}_h - \mathbf{C}_{h,H}) = \operatorname{range}(\mathbf{I}_H^h(\mathbf{A}_H)^{-1}\mathbf{I}_h^H\mathbf{A}_h) = \operatorname{range}(\mathbf{I}_H^h),$$

which concludes the proof. □

The following theorem states the principal result that motivates the use of the Galerkin coarse-grid system operator, that is, the variational principle discussed above.

Theorem 2.6.1. *Suppose that $\mathbf{A}_h$ is symmetric positive definite, $\mathbf{I}_H^h$ is a full rank interpolation operator, and $\mathbf{C}_{h,H}$ is the Galerkin-based coarse-grid correction operator. Then,*

$$\|\mathbf{C}_{h,H}\mathbf{e}_h\|_{\mathbf{A}_h} = \min_{\mathbf{e}_H}\|\mathbf{e}_h - \mathbf{I}_H^h\mathbf{e}_H\|_{\mathbf{A}_h} \quad \textit{for all} \;\; \mathbf{e}_h.$$

Proof. By Proposition 2.6.1,

$$\text{range}(\mathbf{C}_{h,\,H}) \perp_{\mathbf{A}_h} \text{range}(\mathbf{I}_h - \mathbf{C}_{h,\,H}) \quad \text{and} \quad \text{range}(\mathbf{I}_h - \mathbf{C}_{h,\,H}) = \text{range}(\mathbf{I}_H^h).$$

Therefore, by writing $\mathbf{e}_h = \mathbf{C}_{h,\,H}\mathbf{e}_h + (\mathbf{I}_h - \mathbf{C}_{h,\,H})\mathbf{e}_h$, it follows for any fine-grid error $\mathbf{e}_h$ that,

$$\begin{aligned}
\min_{\mathbf{e}_H} \|\mathbf{e}_h - \mathbf{I}_H^h\mathbf{e}_H\|_{\mathbf{A}_h}^2 &= \min_{\mathbf{v}_h \in \text{range}(\mathbf{I}_h - \mathbf{C}_{h,\,H})} \|\mathbf{C}_{h,\,H}\mathbf{e}_h + (\mathbf{I}_h - \mathbf{C}_{h,\,H})\mathbf{e}_h - \mathbf{v}_h\|_{\mathbf{A}_h}^2 \\
&= \min_{\hat{\mathbf{v}}_h \in \text{range}(\mathbf{I}_h - \mathbf{C}_{h,\,H})} \|\mathbf{C}_{h,\,H}\mathbf{e}_h + \hat{\mathbf{v}}_h\|_{\mathbf{A}_h}^2 \\
&= \min_{\hat{\mathbf{v}}_h \in \text{range}(\mathbf{I}_h - \mathbf{C}_{h,\,H})} \left(\|\mathbf{C}_{h,\,H}\mathbf{e}_h\|_{\mathbf{A}_h}^2 + \|\hat{\mathbf{v}}_h\|_{\mathbf{A}_h}^2\right) \\
&= \|\mathbf{C}_{h,\,H}\mathbf{e}_h\|_{\mathbf{A}_h}^2.
\end{aligned}$$

□

Now consider principle **P1**. Naturally, our first step is to characterize the smooth error of simple relaxation schemes in an algebraic setting. Recall from §2.6.1 that in the geometric setting, the most important property of smooth error is that it is not effectively reduced by standard relaxation methods (see Figure 2.9). As discussed in [19], such error can typically be approximated by the *near nullspace* of $\mathbf{A}$, which is the subspace spanned by eigenvectors of $\mathbf{A}$ corresponding to small eigenvalues. In the algebraic setting, error for which relaxation is slow to converge is defined as *algebraically smooth* to distinguish it from the geometric case in which case smooth error is also geometrically smooth. In fact, algebraically smooth error can still be largely oscillatory. Given the iteration matrix of a relaxation scheme, $\mathbf{H} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$, algebraically smooth error is characterized by

$$\|\mathbf{He}\|_{\mathbf{A}} \approx \|\mathbf{e}\|_{\mathbf{A}}. \tag{2.51}$$

Observing that

$$\mathbf{He} = \mathbf{e} - \mathbf{M}^{-1}\mathbf{r},$$

it follows that convergence must be slow if the residuals are in some sense small compared with the errors. Moreover, the converse of this statement is typically true for many commonly used relaxation methods, that is, slow convergence implies small residuals. In [19] Brandt shows that for any symmetric M-matrix $\mathbf{A}$ (more generally for any symmetric positive definite matrix $\mathbf{A}$), if the convergence speed of weighted Jacobi or Gauss–Seidel relaxation slows down, then

$$\sum_{i=1}^{n} \frac{{r_i}^2}{a_{ii}} \ll \sum_{i=1}^{n} r_i e_i, \tag{2.52}$$

where r_i is the ith entry of the residual vector $\mathbf{r} = \mathbf{Ae}$. This result implies that, on average, for each i, algebraically smooth error has scaled residuals that are much smaller than the error itself

$$|r_i| \ll a_{ii}|e_i|. \tag{2.53}$$

Although the error may still be large globally, (2.53) implies that

$$r_i \approx 0 \quad \Rightarrow \quad a_{ii}e_i \approx -\sum_{j\in\mathcal{N}_i} a_{ij}e_j. \tag{2.54}$$

Hence, algebraically smooth error e_i can be approximated locally by an average of its neighboring error values e_j. In addition, if $\mathbf{A}$ has nonnegative row sums then a more intuitive definition of algebraically smooth error is possible. We begin by writing

$$\langle \mathbf{Ae}, \mathbf{e}\rangle = \sum_i \sum_{i<j} |a_{ij}|(e_i - e_j)^2 + \sum_i \sum_j a_{ij}{e_i}^2. \tag{2.55}$$

It follows by the Cauchy–Schwarz inequality with $\mathbf{D} = \mathrm{diag}(\mathbf{A})$ that

$$\langle \mathbf{Ae}, \mathbf{e}\rangle = \langle \mathbf{D}^{-1/2}\mathbf{Ae}, \mathbf{D}^{1/2}\mathbf{e}\rangle \leq \|\mathbf{D}^{-1/2}\mathbf{Ae}\|_2 \|\mathbf{D}^{1/2}\mathbf{e}\|_2 = \|\mathbf{r}\|_{\mathbf{D}^{-1}}\|\mathbf{e}\|_{\mathbf{D}}. \tag{2.56}$$

In conjunction with (2.52) this result implies that $\langle \mathbf{Ae}, \mathbf{e}\rangle \ll \langle \mathbf{De}, \mathbf{e}\rangle$. It now follows from

(2.55) and weak row diagonal dominance of $\mathbf{A}$ that

$$\sum_i \sum_{i<j} |a_{ij}|(e_i - e_j)^2 \ll \sum_i a_{ii} {e_i}^2.$$

Rearranging this expression, we obtain that in general for each i,

$$\sum_{j<i} \frac{|a_{ij}|}{a_{ii}} \frac{(e_i - e_j)^2}{{e_i}^2} \ll 1. \tag{2.57}$$

This expression leads us to one of the main heuristics in classical AMG.

AMG Heuristic: Algebraically smooth error varies slowly in the direction of large (negative) connections. That is, if $|a_{ij}|/a_{ii}$ is relatively large then $e_i \approx e_j$, in which case the error is locally almost constant.

To quantify "large negative connections" in the matrix $\mathbf{A}$, the concept of *strength of connection* is introduced.

Definition 2.6.2 (Strength of connection)**.** Let $\mathbf{A}$ be an M-matrix. Given a threshold value $0 < \theta \leq 1$, the variable u_i (point i) *strongly depends* on variable u_j (point j) if

$$-a_{ij} \geq \theta \max_{k \neq i}\{-a_{ik}\}.$$

Moreover, if u_i strongly depends on u_j, then we say that u_j *strongly influences* u_i.

Therefore, we say that *algebraically smooth error varies slowly in the direction of these strong connections*. By the AMG Heuristic algebraically smooth error is geometrically smooth along strong connections; consequently, it can be accurately interpolated from error points (coarse-grid points) on which it strongly depends. We note that positive off-diagonal connections are considered to be *weak* connections by Definition 2.6.2. Moreover, it is possible that a point i strongly depends on j, but that j only weakly depends on i, even though $\mathbf{A}$ is symmetric.

Now suppose that the current fine grid $\Omega_h = \{1, \ldots, n_h\}$ has been partitioned into a set of coarse points $\mathcal{C}$, and a set of fine points $\mathcal{F}$. Points in $\mathcal{C}$ are referred to as C-points and points in $\mathcal{F}$ as F-points. We note that the coarse grid Ω_H is obtained by relabeling the elements of $\mathcal{C} = \{i_1, \ldots, i_{n_H}\}$ in terms of their coarse-grid labels. For any fine-grid point $i_k \in \mathcal{C}$, its coarse-grid label is given by $\alpha(i_k) = k$. The $n_h \times n_H$ interpolation operator $\mathbf{I}_H^h$ is then defined by

$$(\mathbf{I}_H^h)_{i,\alpha(j)} = \begin{cases} w_{ij} & \text{if } i \in \mathcal{F} \text{ and } j \in \mathcal{C}_i, \\ 1 & \text{if } i \in \mathcal{C} \text{ and } j = i, \\ 0 & \text{otherwise,} \end{cases} \tag{2.58}$$

where the w_{ij}s are the interpolation weights, and $\mathcal{C}_i \subset \mathcal{C}$ is the subset of C-points that strongly influence point i. It remains to determine the interpolation weights w_{ij} for each point $i \in \mathcal{F}$. Let $\mathcal{S}_j$ be the set of points that strongly influence j, for each point $j \in \Omega_h$. Recall that algebraically smooth error is characterized by small residuals, $\mathbf{r} \approx \mathbf{0}$, which implies that

$$a_{ii}e_i \approx -\sum_{j \in \mathcal{N}_i} a_{ij}e_j.$$

Note that for ease of notation we drop the subscript h on the elements of $\mathbf{A}_h$. Suppose the direct neighborhood of i is partitioned by $\mathcal{N}_i = \mathcal{C}_i \cup \mathcal{D}_i^s \cup \mathcal{D}_i^w$, where $\mathcal{D}_i^s = \mathcal{F} \cap \mathcal{S}_i$, and $\mathcal{D}_i^w$ is the set of all points weakly connected to i. Splitting up the summation over the partition of $\mathcal{N}_i$, it follows that

$$a_{ii}e_i \approx -\sum_{j \in \mathcal{C}_i} a_{ij}e_j - \sum_{j \in \mathcal{D}_i^s} a_{ij}e_j - \sum_{j \in \mathcal{D}_i^w} a_{ij}e_j. \tag{2.59}$$

Observe that if $\mathcal{D}_i^s = \mathcal{D}_i^w = \emptyset$, then (2.59) would give the desired interpolation formula with weights $w_{ij} = -a_{ij}/a_{ii}$ for all $j \in \mathcal{C}_i$. This observation suggests that unwanted connections in $\mathcal{D}_i^s \cup \mathcal{D}_i^w$ should be "collapsed" to $\mathcal{C}_i$. For weakly connected neighbors of point i, that is, for points in $\mathcal{D}_i^w$, each e_j is replaced by e_i. More care, however, must be taken with the strong connections in $\mathcal{D}_i^s$. Let j belong to $\mathcal{D}_i^s$. A premise of AMG is that smooth error is locally almost constant along strong connections. Assuming that $\mathcal{C}_i \cap \mathcal{S}_j \neq \emptyset$, it follows that e_j can be approximated by a convex combination of e_k for $k \in \mathcal{C}_i$ that interpolates constants exactly. By the small residual condition coefficients in the convex combination are proportional to connections between j and each point $k \in \mathcal{C}_i$:

$$e_j \approx -\frac{1}{a_{jj}} \sum_{k \in \mathcal{N}_j} a_{jk} e_k \approx -\frac{1}{a_{jj}} \sum_{k \in \mathcal{C}_i} a_{jk} e_k.$$

Normalizing so that constants are interpolated exactly,

$$e_j \approx \frac{\sum_{k \in \mathcal{C}_i} a_{jk} e_k}{\sum_{k \in \mathcal{C}_i} a_{jk}}. \tag{2.60}$$

Clearly, the approximation in (2.60) improves as the amount overlap between $\mathcal{C}_i$ and $\mathcal{S}_j$ increases. At the very minimum, however, these sets must have at least one point in common. We note that (2.60) can be thought of as a truncated interpolation formula for e_j, in the sense that we have simply deleted those points in the neighborhood of j that do not belong to $\mathcal{C}_i$. Substituting these approximations into (2.59) and solving for e_i leads to the following formula for the interpolation weights:

$$w_{ij} = -\frac{a_{ij} + \sum_{m \in \mathcal{D}_i^s} \left(\frac{a_{im} a_{mj}}{\sum_{k \in \mathcal{C}_i} a_{mk}} \right)}{a_{ii} + \sum_{r \in \mathcal{D}_i^w} a_{ir}} \quad \text{for all } \; j \in \mathcal{C}_i. \tag{2.61}$$

In our description of AMG interpolation we assumed that the set of coarse-grid points $\mathcal{C}$ was available. We now describe how $\mathcal{C}$ is determined. Let $\mathcal{S}_i^\top$ denote the set of points that strongly depend on point i. The coarsening process seeks to satisfy two heuristics:

H1: For each point $i \in \mathcal{F}$, every point $j \in \mathcal{S}_i$ should either belong to $\mathcal{C}_i$ or should strongly depend on at least one point in $\mathcal{C}_i$.

H2: The set of coarse-grid points $\mathcal{C}$ should be a maximal subset of all points with the property that no two points in $\mathcal{C}$ are strongly connected to each other.

Heuristic **H1** (see Figure 2.14) is essential if approximation (2.60) in the interpolation formula is to make sense. Heuristic **H2** seeks to strike a balance between the amount of work on the coarse grid and the convergence speed of the multigrid cycle, thereby controlling the efficiency of the solution process. Because it is not always possible to satisfy both heuristics, the coarsening process seeks to rigorously satisfy **H1**, while using **H2** as a guide.

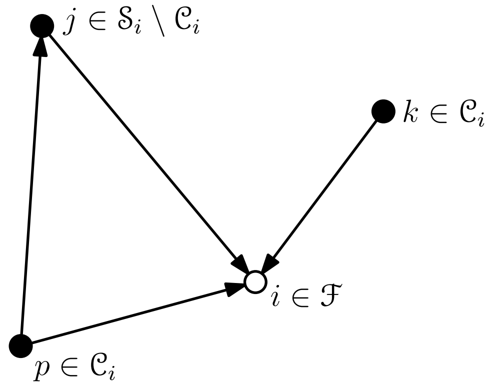


Figure 2.14: Illustration of coarsening heuristic **H1**. Arrows indicate the direction of strong influence.

Selection of the C-points proceeds in two phases. In the first phase (Algorithm 2.3), an initial partition of Ω_h into sets $\mathcal{C}$ and $\mathcal{F}$ is computed. The goal of this phase is obtain a uniform distribution of C-points over Ω_h that tend to satisfy **H2**. As shown by Algorithm 2.3, associated with each unassigned point i is a measure, λ_i, of that point's usefulness as a C-point. Intuitively, points that influence a large number of F-points, that is, points with large measures, represent good candidates for C-points. During each iteration of Algorithm 2.3, the point with maximal measure is added to $\mathcal{C}$, and the unassigned points that strongly depend on this point are added to $\mathcal{F}$. Unassigned points that influence the new F-points are more likely to be useful as C-points, so their measures are incremented. On line 5 of Algorithm 2.3, λ_j is decremented for all unassigned points $j \in \mathcal{S}_i$ to try to minimize the number of strong C-C connections.

The second phase of the coarsening process (Algorithm 2.4) seeks to rigorously enforce heuristic **H1**. The algorithm sequentially tests each F-point i to ensure that each point in $\mathcal{D}_i^s$ strongly depends on at least one point in $\mathcal{C}_i$. If for some F-point i, a point $j \in \mathcal{D}_i^s$

is found such that $\mathcal{C}_i \cap \mathcal{S}_j = \emptyset$, then j is tentatively made a C-point and is added to $\mathcal{C}_i$. Processing of the points in $\mathcal{D}_i^s$ then continues. If all those points now strongly depend on $\mathcal{C}_i$, then j is removed from $\mathcal{F}$ and is permanently added to $\mathcal{C}$. However, if another point in $\mathcal{D}_i^s$ is found that does not depend on any points in $\mathcal{C}_i$, then i is permanently added to $\mathcal{C}$ and j is removed from the tentative C-point list. This process is repeated for the next F-point and continues until all F-points have been examined.

Algorithm 2.3: Phase 1 of AMG coarse-grid selection.

```
1. Set $\mathcal{C} \leftarrow \emptyset$, $\mathcal{F} \leftarrow \emptyset$, $\mathcal{U} \leftarrow \Omega_h$, $\lambda_i \leftarrow |\mathcal{S}_i^\top|$ for all $i \in \mathcal{U}$
   while $\mathcal{U} \neq \emptyset$ do
2.     Select $i \in \mathcal{U}$ with maximal $\lambda_i$ and set $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$, $\mathcal{U} \leftarrow \mathcal{U} \setminus \{i\}$
       foreach $j \in (\mathcal{S}_i^\top \cap \mathcal{U})$ do
3.         Set $\mathcal{F} \leftarrow \mathcal{F} \cup \{j\}$ and $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j\}$
4.         For each point $k \in \mathcal{S}_j \cap \mathcal{U}$ set $\lambda_k \leftarrow \lambda_k + 1$
       end
5.     For each $j \in \mathcal{S}_i \cap \mathcal{U}$ set $\lambda_j \leftarrow \lambda_j - 1$
   end
```

Algorithm 2.4: Phase 2 of AMG coarse-grid selection.

```
1. Set $\mathcal{T} \leftarrow \mathcal{F}$
   while $\mathcal{T} \neq \emptyset$ do
2.     Select a point $i \in \mathcal{T}$ and set $\mathcal{C}_i \leftarrow \mathcal{C} \cap \mathcal{S}_i$, $\tilde{\mathcal{C}}_i \leftarrow \emptyset$, $\mathcal{D}_i^s \leftarrow \mathcal{S}_i \setminus \mathcal{C}_i$, $\mathcal{T} \leftarrow \mathcal{T} \setminus \{i\}$
       foreach $j \in \mathcal{D}_i^s$ such that $\mathcal{C}_i \cap \mathcal{S}_j = \emptyset$ do
           if $\tilde{\mathcal{C}}_i \neq \emptyset$ then
3.             Set $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$, $\mathcal{F} \leftarrow \mathcal{F} \setminus \{i\}$, and go to line 2
           else
4.             Set $\tilde{\mathcal{C}}_i \leftarrow \{j\}$, $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{j\}$
           end
       end
5.     Set $\mathcal{C} \leftarrow \mathcal{C} \cup \tilde{\mathcal{C}}_i$, $\mathcal{F} \leftarrow \mathcal{F} \setminus \tilde{\mathcal{C}}_i$, $\mathcal{T} \leftarrow \mathcal{T} \setminus \tilde{\mathcal{C}}_i$
   end
```

The application of AMG is usually implemented as a two part process composed of a *setup phase* and a *solution phase*. The setup phase is responsible for choosing the coarse grids, and defining the transfer and coarse-grid system operators. The solution phase consists of repeated multigrid cycles until the desired solution accuracy is achieved. Unlike geometric multigrid where the numerical work per cycle is typically known beforehand, in AMG nothing definite is known a priori about the numerical work per cycle. As discussed in [103], the total work for both the setup and solution phases can be estimated in terms of the following quantities:

1. The average "stencil size" over all grids
2. The average number of interpolation points per F-point
3. The ratio of the total number of points on all grids to that of the finest grid, referred to as the *grid complexity* (C_{grid})
4. The ratio of the total number of nonzero elements in all system operators to that of the finest-grid system operator, referred to as the *operator complexity* (C_{op})

In an efficient multigrid solver these quantities should all be relatively small. The grid complexity and the operator complexity are useful measures of the memory requirements for storing the solution vectors and right-hand side vectors, and the system operators on all levels, respectively. With respect to the time complexity of an AMG method, the most significant quantities are the stencil size and the operator complexity. The numerical work of a multigrid cycle (solve phase) is dominated by the relaxation work on all levels, which is proportional to the number of nonzero elements in all matrices on all levels. Therefore, the operator complexity is an approximate measure of the ratio of numerical work per cycle to the amount of relaxation work on the finest grid. Small operator complexities lead to small cycle times. The *stencil size* on a given level is defined as the average number of coefficients per matrix row. While the stencil size of the original matrix is usually small, large stencil sizes are possible on coarser levels. The stencil size largely influences the setup time, in that stencil growth can lead to a significant increase in the number of operations required by coarsening and interpolation processes [48]. Naturally, stencil growth also leads to increased operator complexities.

The numerical efficiency of a multigrid method depends on both the amount of work per cycle and the speed of convergence. The speed of convergence is determined by the asymptotic convergence factor ρ (see (2.32) in §2.5). It is useful to have an empirical measurement of this quantity. The convergence factor of the kth cycle, $q^{(k)}$, is defined as

$$q^{(k)} := \frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{r}^{(k-1)}\|},$$

where $\mathbf{r}^{(j)}$ is the residual of the updated solution after the jth cycle. For sufficiently large k, a good estimate for ρ is then given by

$$\gamma^{(k)} := \sqrt[k-k_0]{q^{(k_0+1)} \dots q^{(k)}} = \sqrt[k-k_0]{\frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{r}^{(k_0)}\|}}, \tag{2.62}$$

which is the average residual reduction factor over the (k_0+1)th to the kth cycles. We note that because the first few iterations tend not to reflect the asymptotic convergence factor a small value of k_0 is used, usually between two and five. Clearly, $\gamma^{(k)} \ll 1$ corresponds to a rapid reduction of the residual. A *numerically scalable* algorithm is one in which the amount of computational work to solve a problem of a given size is linearly related to the problem size. Numerical scalability of the AMG solution phase is characterized by asymptotically constant operator complexities and convergence factors as the problem size grows. In practice, linear scaling of the setup phase can typically be achieved for sparse matrices whose stencil sizes do not grow too large on the coarser grids. While the setup phase is more expensive than the solution cycles due to construction of the transfer operators and coarse-grid system operators, its numerical cost can often be amortized over many runs.

2.6.4 Aggregation-based algebraic multigrid

The reduction of error components by a correction based on grouping states together dates back to the work of Southwell [108]. The concept of grouping nodes into disjoint sets and the subsequent identification of each set with a single degree of freedom was introduced in the early 1950s by Leontief [84]. Iterative Aggregation-based methods have been considered at least as far back as the 1982 paper by Chatelin and Miranker [37], which explored this approach in the context of Markov chains. True AMG methods using aggregation were first considered by Vaněk [121] and Braess [17]. The main difference between the classical AMG framework described in §2.6.3 and aggregation-based AMG is the definition of coarsening and interpolation. In the latter, the current set of fine-grid points Ω_h is partitioned into mutually disjoint sets called "aggregates" (as opposed to a C/F-splitting)

$$\Omega_h = \{\mathcal{A}_p : p = 1, \ldots, n_H\},$$

where each aggregate $\mathcal{A}_p$ may be associated with a coarse-grid point $p \in \Omega_H = \{1, \ldots, n_H\}$. Interpolation is then defined by *piecewise constant interpolation* from Ω_H to the aggregates, that is, the interpolant $\tilde{\mathbf{u}}_h$ of an arbitrary coarse-level vector $\mathbf{u}_H$ is given by

$$(\tilde{\mathbf{u}}_h)_i = (\mathbf{u}_H)_p \quad \text{for all} \quad i \in \mathcal{A}_p.$$

It is easy to see that aggregation-based AMG can be interpreted as a simple limiting case of the classical AMG approach, in which each F-point is allowed to interpolate from exactly one C-point. That is, although an F-point i may have multiple connections to the set of C-points, its coarse interpolatory set is restricted to contain exactly one C-point. From this point of view each aggregate consists of exactly one C-point, $k \in \mathcal{C}$, as well as the indices of all F-points that interpolate from k (see Figure 2.15). Clearly, for this partitioning into aggregates to be reasonable, the F-points in each aggregate should all strongly depend on the corresponding interpolatory C-point. Setting all interpolation weights equal to unity, it follows from (2.58) that classical AMG interpolation then corresponds to piecewise constant interpolation.

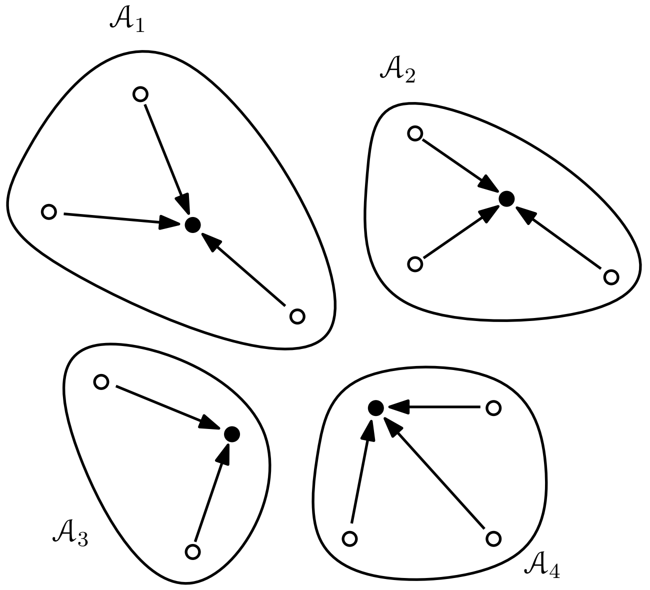


Figure 2.15: Partition of fine-grid points into aggregates (• C-point, ∘ F-point). The arrows indicate which C-point an F-point interpolates from.

Let us now consider the aggregation process in slightly more detail, and show how it relates to the classical AMG approach. Suppose that Ω_h has been partitioned into aggregates, and let the current $n_h \times n_h$ fine-grid problem be given by

$$\mathbf{A}_h \mathbf{u}_h = \mathbf{f}_h \quad \text{with} \quad \mathbf{u}_h, \mathbf{f}_h \in \mathbb{R}^{n_h}.$$

Then the $n_H \times n_H$ coarse-grid problem

$$\mathbf{A}_H \mathbf{u}_H = \mathbf{f}_H \quad \text{with} \quad \mathbf{e}_H, \mathbf{f}_H \in \mathbb{R}^{n_H}$$

is obtained through a simple summing process, where the entries of the Galerkin coarse-grid system operator are given by

$$(\mathbf{A}_H)_{pq} = \sum_{i \in \mathcal{A}_p} \sum_{j \in \mathcal{A}_q} (\mathbf{A}_h)_{ij}, \tag{2.63}$$

and

$$(\mathbf{f}_H)_p = \sum_{i \in \mathcal{A}_p} (\mathbf{f}_h)_i \tag{2.64}$$

for $p, q = 1, \ldots, n_H$. Defining the *disaggregation operator* (interpolation operator) by

$$(\mathbf{Q}_H^h)_{ip} = \begin{cases} 1 & \text{if } i \in \mathcal{A}_p, \\ 0 & \text{otherwise} \end{cases} \tag{2.65}$$

and the *aggregation operator* (restriction operator) by $\mathbf{Q}_h^H = (\mathbf{Q}_H^h)^\top$, expressions (2.63) and (2.64) may be written as

$$\mathbf{A}_H = \mathbf{Q}_h^H \mathbf{A}_h \mathbf{Q}_H^h \quad \text{and} \quad \mathbf{f}_H = \mathbf{Q}_h^H \mathbf{f}_h. \tag{2.66}$$

Thus, the aggregation AMG approach is equivalent to the classical AMG approach with the only difference being the construction of the interpolation operator.

In that many aggregation strategies are possible, we mention a general purpose aggregation strategy referred to as *neighborhood aggregation* [123] that follows a two-phase procedure similar to classical AMG coarsening. In the first phase, aggregates are constructed by selecting a root point $i \in \Omega_h$ that is not adjacent to any other aggregate, and including all points that are strongly connected to i (based on some strength of connection measure) including i itself. This procedure is repeated until all unaggregated points are adjacent to an aggregate. In the second phase, the remaining unaggregated points are either integrated into preexisting aggregates, or are combined to form new aggregates. Considerable care must be taken in handling the remaining unaggregated points during the second phase. If too many aggregates are formed, then coarsening will be slow and cycle times may increase. If aggregates are enlarged by too much, or have largely varying sizes or irregular shapes, then convergence factors will grow. We note that the neighborhood aggregation process is described in more detail in §4.1.

The disaggregation operator defined in (2.65) possesses a number of useful properties that we now briefly discuss.

Proposition 2.6.2. *If the fine-grid operator $\mathbf{A}_h$ is a symmetric M-matrix with zero row sums, then so is the coarse-grid system operator $\mathbf{A}_H$ given by* (2.63).

Proof. According to Theorem 2.2.7 (4) it is sufficient to show that $\mathbf{A}_H$ is a symmetric positive definite matrix with nonpositive off-diagonal elements and zero row sums. The fact that $\mathbf{A}_H$ is SPD follows directly from the observation that $\mathbf{Q}_H^h$ has full rank. The off-diagonal elements of $\mathbf{A}_H$ are nonpositive because the aggregates do not overlap, that is, because $\mathcal{A}_p \cap \mathcal{A}_q = \emptyset$ for all $p \neq q$ the summation in (2.63) is only over the off-diagonal elements of $\mathbf{A}_h$. The zero row sum property follows by the fact that $\mathbf{Q}_H^h$ interpolates constants exactly, that is, $\mathbf{A}_H \mathbf{1}_H = \mathbf{Q}_h^H \mathbf{A}_h \mathbf{Q}_H^h \mathbf{1}_H = \mathbf{Q}_h^H \mathbf{A}_h \mathbf{1}_h = \mathbf{0}_h$. □

Owing to the simple structure of the disaggregation operator, storing $\mathbf{Q}_{\ell+1}^{\ell}$ requires only n_ℓ storage locations on level ℓ, and hence only nC_{grid} storage locations for a multilevel V-cycle. Moreover, restriction (interpolation) of a fine-grid (coarse-grid) vector requires only n flops, and constructing the aggregated coarse-grid matrix $\mathbf{A}_H$ requires no more than $n_H \, \mathrm{nnz}(\mathbf{A}_h)$ flops. Perhaps the most important property of the aggregation process is that the Galerkin construction of the coarse-grid system operator approximately preserves the number of nonzero elements per row of the fine-grid matrix [62]. Consequently, aggregation multigrid usually exhibits little, if any, stencil growth, which leads to small operator complexities ($C_{op} \approx 1$) and fast cycles.

Unfortunately, the simple aggregation procedure described above typically leads to inefficient AMG methods that are slow to converge. However, with certain modifications aggregation-based AMG can become practical. It was shown independently by Blaheta and Vaněk [13, 125] how weighting the coarse-grid correction by a small scalar factor $\alpha > 1$, that is,

$$\mathbf{u}_h^{\text{CGC}} = \bar{\mathbf{u}}_h + \alpha \mathbf{Q}_H^h \mathbf{v}_H,$$

can significantly improve the convergence of basic aggregation-based AMG. This procedure, referred to as *over-correction*, was motivated by the observation that while the correction typically approximates the error well in the sense of its “progress”, it may not provide a good approximation in the sense of its “size” [125]. In the case of symmetric positive definite matrices, theoretical arguments are used to derive a formula for the optimal over-correction parameter α that minimizes the error of the CGC in the energy norm. For further detail we refer to Chapter 5, where over-correction is more fully introduced, and is used to accelerate aggregation-based AMG methods for Markov chains.

Another approach to accelerate aggregation-based AMG is the so-called smoothed aggregation (SA) method developed in [121, 124]. The SA method differs from the basic aggregation approach primarily in its definition of the interpolation operators. Observe that interpolation of a coarse-grid vector $\mathbf{v}_H$ can be written as

$$\mathbf{Q}\mathbf{v}_H = \sum_{p=1}^{n_H} (\mathbf{v}_H)_p \, \boldsymbol{\varphi}_p,$$

where $\boldsymbol{\varphi}_j$ is the jth column of $\mathbf{Q}_H^h$. Therefore, interpolation can be interpreted as a linear combination of basis vectors $\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_{n_H}$. Moreover, because $\boldsymbol{\varphi}_p$ has support given by the corresponding aggregate $\mathcal{A}_p$, for $p = 1, \ldots, n_H$, the basis vectors form an orthogonal basis for the range of interpolation. Thus, the aggregation-based AMG framework for defining interpolation can be viewed as an approach to define appropriate (locally supported) basis vectors that span the near nullspace of $\mathbf{A}_h$ [72]. The SA method embraces this viewpoint by using a smoother such as weighted Jacobi or Gauss–Seidel to smooth the columns of $\mathbf{Q}_H^h$, thereby creating a set of smoothed basis vectors and introducing overlap between the aggregates. The smoothed basis vectors can more accurately approximate the near-nullspace components of $\mathbf{A}_h$ (for example see [119]), and the SA multigrid method typically performs similarly to classical AMG. Unfortunately, the desirable properties of $\mathbf{Q}_H^h$ such as those given by Proposition 2.6.2 may be lost through the smoothing process, and in particular, operator complexities may grow.

2.6.5 Full approximation scheme and full multigrid

The full approximation scheme (FAS) [18, 32, 119] is a multigrid method intended for solving nonlinear problems. It is based on the same fundamental principles as in the linear case, that is, pre- and post-smoothing of the fine-grid error combined with a coarse-grid correction procedure. The main difference between FAS and linear multigrid methods is that in FAS the *full approximation* is approximated on the coarse grid, instead of the error. Naturally, this difference is how the full approximation scheme earned its name.

Consider a system of nonlinear algebraic equations given by

$$\mathbf{A}(\mathbf{u}) = (A_1(\mathbf{u}), \dots, A_n(\mathbf{u}))^\top = (f_1, \dots, f_n)^\top = \mathbf{f}, \tag{2.67}$$

with functionals $A_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \dots, n$. Common relaxation schemes for this type of problem include the nonlinear Jacobi and nonlinear Gauss–Seidel methods. During the kth iteration of the nonlinear Jacobi method, the ith variable is updated by solving the ith equation of (2.67) for $u_i^{(k+1)}$, that is,

$$A_i(u_1^{(k)}, \dots, u_{i-1}^{(k)}, u_i^{(k+1)}, u_{i+1}^{(k)}, \dots, u_n^{(k)}) - f_i = 0.$$

Analogously, during the kth iteration of the nonlinear Gauss–Seidel method, the ith variable is updated by solving

$$A_i(u_1^{(k+1)}, \dots, u_{i-1}^{(k+1)}, u_i^{(k+1)}, u_{i+1}^{(k)}, \dots, u_n^{(k)}) - f_i = 0$$

for $u_i^{(k+1)}$. In both cases, a single nonlinear equation has to be solved for a single unknown. Updating the ith variable is then formally equivalent to finding a root of a nonlinear function of a single variable. Therefore, relaxation methods for nonlinear problems are usually combined with a root-finding method such as scalar Newton's method, resulting in, for example, the Jacobi–Newton method and the Gauss–Seidel–Newton method. Although these methods may be inefficient solvers for the system of nonlinear equations (2.67), they are often effective smoothers.

Similar to the linear case, a FAS cycle can be defined by recursively extending a two-grid method. Thus, for simplicity we describe one iteration of a two-grid FAS cycle. Let the fine-grid problem be given by

$$\mathbf{A}_h(\mathbf{u}_h) = \mathbf{f}_h,$$

and suppose there exist inter-grid transfer operators $\mathbf{I}_H^h$ and $\mathbf{I}_h^H$. Furthermore, let $\bar{\mathbf{u}}_h$ be the smoothed fine-grid approximation, and let $\mathbf{e}_h = \mathbf{u}_h - \bar{\mathbf{u}}_h$ be the corresponding error. Then the fine-grid residual equation is given by

$$\mathbf{A}_h(\bar{\mathbf{u}}_h + \mathbf{e}_h) - \mathbf{A}_h(\bar{\mathbf{u}}_h) = \mathbf{f}_h - \mathbf{A}_h(\bar{\mathbf{u}}_h) = \mathbf{r}_h. \tag{2.68}$$

This equation is approximated on the coarse grid by

$$\mathbf{A}_H(\bar{\mathbf{u}}_H + \mathbf{e}_H) - \mathbf{A}_H(\bar{\mathbf{u}}_H) = \mathbf{r}_H = \mathbf{I}_h^H(\mathbf{f}_h - \mathbf{A}_h(\bar{\mathbf{u}}_h)), \tag{2.69}$$

where $\mathbf{A}_H$ is an appropriate nonlinear operator. For example, in the case of a nonlinear boundary value problem, $\mathbf{A}_H$ may be obtained by discretizing the original problem on the coarse grid. On the coarse grid we work with the problem

$$\mathbf{A}_H(\mathbf{w}_H) = \mathbf{f}_H, \tag{2.70}$$

where $\mathbf{w}_H = \bar{\mathbf{u}}_H + \mathbf{e}_H$ is the full approximation (not the error) and

$$\mathbf{f}_H \coloneqq \mathbf{A}_H(\mathbf{I}_h^H\bar{\mathbf{u}}_h) + \mathbf{I}_h^H(\mathbf{f}_h - \mathbf{A}_h(\bar{\mathbf{u}}_h)). \tag{2.71}$$

The relaxed fine-grid solution is transfered to the coarse grid via restriction to obtain $\bar{\mathbf{u}}_H = \mathbf{I}_h^H\bar{\mathbf{u}}_h$. Solving (2.70) for $\mathbf{w}_H$, the coarse-grid approximation of the error is given by $\mathbf{e}_H = \mathbf{w}_H - \mathbf{I}_h^H\mathbf{u}_h$. We note that on the coarsest grid it may be necessary to solve the linearized version of (2.70) to obtain an approximation of $\mathbf{w}_H$. The coarse-grid corrected solution is then computed by

$$\mathbf{u}_h^{\text{CGC}} = \bar{\mathbf{u}}_h + \mathbf{I}_H^h(\mathbf{w}_H - \mathbf{I}_h^H\mathbf{u}_h). \tag{2.72}$$

The two-grid scheme concludes with a few post-smoothing iterations.

A few observations regarding FAS are in order. We note that if $\mathbf{A}$ is a linear operator, then FAS reduces to the usual linear two-grid correction scheme. Thus, the full approximation scheme can be viewed as a generalization of the two-grid correction scheme for linear problems (see Algorithm 2.1). Also, if the exact fine-grid solution $\mathbf{u}_h$ is a fixed point of the relaxation scheme and $\mathbf{I}_h^H \mathbf{u}_h$ is a fixed point of the coarse-grid solver, then $\mathbf{u}_h$ is a fixed point of FAS. In practice the coarse-grid equation (2.70) is solved recursively, and so the full approximation scheme is usually implemented as a V-cycle or a W-cycle. For further details on FAS see [26, 119].

As is true for any iterative method, convergence of the full approximation scheme can often be greatly improved by supplying a suitable initial guess. This statement is especially true for nonlinear problems, in which the initial guess must lie within the basin of attraction. Moreover, when the approximate solution is close to an exact solution, the nonlinear problem (2.67) appears more linear, and the solution process is more effective. One way in which we can try to obtain a better initial guess to the fine-grid problem is to use full multigrid (FMG) [32, 119]. Full multigrid is based on the idea of *nested iterations*, whereby coarse grids are used to obtain improved initial guesses for fine-grid problems. At any given grid level the problem is first solved on the next coarser grid, after which the solution is interpolated to the current grid to provide a good initial guess. This process naturally starts at the coarsest grid and terminates at the finest. Once an initial guess to the finest-grid problem has been obtained, repeated multigrid cycles can be used to obtain an accurate solution. We note that the use of full multigrid is not restricted to nonlinear problems solved by FAS; FMG can also provide a good initial guess for linear problems in which geometric multigrid or AMG is used. The following components are required to define an FMG cycle:

1. a sequence of consecutively coarser grids $\Omega_0, \ldots, \Omega_L$,
2. a multigrid solver MGCYC($\mathbf{u}$, $\mathbf{A}$, $\mathbf{f}$, μ, ν_1, ν_2) with its own multigrid hierarchy defined on the same grid structure as in 1,
3. FMG transfer operators $\hat{\mathbf{I}}_{\ell+1}^{\ell}$ and $\hat{\mathbf{I}}_{\ell}^{\ell+1}$ for $\ell = 1, \ldots, L-1$, possibly different from those used by the multigrid solver MGCYC.

A generic full multigrid cycle is given by Algorithm 2.5 (initially with $\ell = 0$). We note that the FMG restriction operators may be the same as those used by the associated multigrid cycle MGCYC, or, alternatively, $\mathbf{f}_\ell$ may be obtained by injection of $\mathbf{f}_0$ to Ω_ℓ.

Algorithm 2.5: FMGCYC (generic full multigrid cycle for solving $\mathbf{Au} = \mathbf{f}$)

```
    Input: $\mathbf{f}_\ell$, $\nu_0 \geq 1$
    Output: $\mathbf{u}_\ell^{\text{FMG}}$
    if $\ell = L$ then
1.      Solve $\mathbf{A}_L\mathbf{u}_L = \mathbf{f}_L$ and set $\mathbf{u}_L^{\text{FMG}} \leftarrow \mathbf{u}_L$
    else
2.      Restrict the right-hand side: $\mathbf{f}_{\ell+1} \leftarrow \hat{\mathbf{I}}_\ell^{\ell+1}\mathbf{f}_\ell$
3.      $\mathbf{u}_{\ell+1}^{\text{FMG}} \leftarrow$ FMGCYC($\mathbf{f}_{\ell+1}, \nu_0$)
4.      Interpolate to the current fine grid: $\mathbf{u}_\ell^{\text{FMG}} \leftarrow \hat{\mathbf{I}}_{\ell+1}^{\ell}\mathbf{u}_{\ell+1}^{\text{FMG}}$
5.      Repeat $\nu_0$ times (usually $\nu_0 = 1$):
            $\mathbf{u}_\ell^{\text{FMG}} \leftarrow$ MGCYC($\mathbf{u}_\ell^{\text{FMG}}$, $\mathbf{A}_\ell$, $\mathbf{f}_\ell$, $\mu$, $\nu_1$, $\nu_2$)
    end
```

2.6.6 Adaptive algebraic multigrid

An efficient multigrid process relies on the appropriate complementarity between relaxation and the coarse-grid correction. To achieve this complementarity, algebraic multigrid makes assumptions about the nature of algebraically smooth error. For example, the classical AMG framework described in §2.6.3 assumes that all algebraically smooth error components vary slowly along strong connections in the fine-grid matrix, that is, they are locally constant. However, in some cases these assumptions are not warranted and error that is not treated effectively by relaxation can vary substantially along strong connections. Adaptive AMG attempts to remedy this situation by reducing or eliminating the method's reliance on these additional assumptions. The main premise of adaptive AMG is to "use the method to improve the method". Essentially, the method should automatically identify troublesome error components and make adjustments for them. The concept of using a multigrid algorithm to improve itself is not new; the key ingredients of adaptive AMG were originally developed in the early stages of the AMG project by Brandt, McCormick, and Ruge [24] (see also [103]). More recently, the concepts of adaptive AMG were developed further in [29, 30, 31].

The basic adaptive algorithm proceeds as follows. Set $k = 0$ and define the initial method, $M^{(k)}$, as the multigrid relaxation scheme. Starting with a random initial guess on the finest grid, apply this method to the homogeneous problem $\mathbf{Au} = \mathbf{0}$ to uncover a representative of algebraically smooth error (referred to as a *prototype*). Using this prototype, define a multigrid hierarchy as follows. At each level apply the current method to the homogeneous problem, and build the interpolation operator and corresponding Galerkin coarse-grid system operator. Interpolation is constructed in such a way that the current prototype lies approximately in its range. Continue this procedure recursively to the coarsest grid. From the coarsest grid, interpolate and relax up to the finest grid. The resulting vector forms the enriched prototype on the finest grid. Define the new method, $M^{(k+1)}$, as the resulting multigrid hierarchy. Test the new method by applying it to $\mathbf{Au} = \mathbf{0}$ with a random initial guess. If its performance is adequate, then the adaptive stage is complete. Otherwise, there remain error components that are not effectively reduced by the current method. If this is the case, the adaptive step is repeated with the new prototype serving as the initial guess, and with k incremented by one.

Thus, the adaptive stage improves the multigrid hierarchy until relaxation and coarse-grid correction are sufficiently complementary. Ideally, the adaptive setup phase should allow for the recovery of classical multigrid performance in cases where the near-nullspace components are not locally constant, albeit at the expense of a more costly setup phase and a more elaborate implementation.

The AMG method we develop for Markov chains (Chapter 3) is related to adaptive AMG in the following way. At each level interpolation is constructed so that its range exactly contains the current relaxed solution. As a consequence, smooth error (the prototype) corresponding to the current solution is approximately represented in the range of interpolation. Therefore, as the computed solution becomes more accurate, the representation of smooth error in the range of interpolation improves, and hence, the coarse representation of the fine-grid problem improves. Thus, the method continually *improves* *itself* as the fine-grid solution converges. In a similar way, the aggregation methods for Markov chains developed by Horton and Leutenegger [69] and by De Sterck et al. [51] are related to adaptive versions of aggregation multigrid [29].

Chapter 3

Markov Chain AMG Method

Having introduced Markov chains and multigrid fundamentals in Chapter 2, we now describe an adaptive AMG method for computing the stationary distribution of irreducible homogeneous Markov chains with finite state spaces. Our approach incorporates classical AMG coarsening and interpolation developed in the early stages of the AMG project by Brandt, McCormick, and Ruge [24] within a multiplicative correction scheme framework. The multiplicative coarse-grid correction process is similar to the two-level aggregated equations proposed by Simon and Ando [107]. The framework is similar to the two-level iterative aggregation/disaggregation method for Markov chains pioneered Takahashi [114], and later extended to the multilevel case by Horton and Leutenegger [69, 85]. Our definition of interpolation is also closely related to the exact interpolation scheme (EIS) proposed by Brandt and Ron [26], which has been applied to various eigenproblems [83, 86]. The EIS is an adaptive multilevel approach in which the interpolation operator on each level is constructed to exactly fit the current approximate solution after pre-relaxation. The coarse level is used to compute a solution, as opposed to an error correction in classical AMG, that upon interpolation to the fine level yields an improved approximation. Moreover, unlike in classical AMG, no residuals are transferred to the coarse level.

With the success of two-level iterative aggregation/disaggregation methods and smoothed aggregation methods for Markov chains [49, 118], AMG is a compelling alternative approach for Markov chains. Algebraic multigrid relies on the twin premises that relaxation produces small residuals, and that the error associated with these residuals is locally constant. These assumptions form the basis for choosing coarse grids and defining interpolation weights. While many linear systems can be treated by relaxation schemes that yield small residuals, the premise that smooth error is locally constant limits the applicability of classical AMG. Indeed, for general nonsymmetric problems such as Markov chains, a rigorous characterization of algebraically smooth error is difficult, if not intractable. Nonetheless, AMG methods have been successfully applied to nonsymmetric problems [31, 38, 105, 112]. As discussed by Stüben in [112], nonsymmetry by itself is typically not a major issue for AMG. In our case, it is reassuring that matrices arising from the study of Markov chains typically share many of the same underlying properties as the matrices for which AMG was developed, for example, sparsity with locally connected graphs, strictly positive diagonal elements, and nonpositive off-diagonal elements.

Let us now consider the problem at hand. Let $\mathbf{B} \in \mathbb{R}^{n\times n}$ be the transition matrix of an irreducible homogeneous Markov chain with finite state space. In the case of a continuous-time Markov chain, $\mathbf{B}$ corresponds to the transition matrix of the uniformized chain (see (2.17) in §2.3). Regardless, $\mathbf{B}$ is an irreducible column-stochastic matrix. The stationary probability vector is the unique vector $\mathbf{x} \in \mathbb{R}^n$ such that

$$\mathbf{x} = \mathbf{B}\mathbf{x}, \qquad x_i > 0 \;\text{ for }\; i = 1, \ldots, n, \qquad \sum_{i=1}^{n} x_i = 1. \tag{3.1}$$

That is, $\mathbf{x}$ is the eigenvector of $\mathbf{B}$ corresponding to the largest eigenvalue. As a matter of convenience we reformulate this eigenproblem as a linear system problem:

$$\mathbf{A}\mathbf{x} = \mathbf{0}, \qquad x_i > 0 \;\text{ for }\; i = 1, \ldots, n, \qquad \sum_{i=1}^{n} x_i = 1, \tag{3.2}$$

where $\mathbf{A} = \mathbf{I} - \mathbf{B}$ is an irreducible singular M-matrix. Subtracting $\mathbf{B}$ from the identity has the effect of reflecting the spectrum of $\mathbf{B}$ across the imaginary axis and then shifting the reflected eigenvalues into the right half plane. Consequently, the eigenvector $\mathbf{x}$ corresponds to the smallest eigenvalue of $\mathbf{A}$. Hence, relaxations applied to $\mathbf{A}\mathbf{x} = \mathbf{0}$ should eliminate the "high eigenvectors" of $\mathbf{A}$ (those eigenvectors corresponding to large eigenvalues), leaving "low eigenvectors" (those eigenvectors corresponding to small eigenvalues) to be handled by coarse-grid correction. Note that we proved convergence of the weighted Jacobi method applied to $\mathbf{A}\mathbf{x} = \mathbf{0}$ in Proposition 2.5.7. In order to obtain an effective method there must exist good *complementarity* between relaxation and coarse-grid correction. Therefore, the range of interpolation must accurately represent the low eigenvectors of $\mathbf{A}$. In particular, because our framework employs a multiplicative correction, the range of $\mathbf{P}$ must accurately represent the exact solution $\mathbf{x}$ (the "lowest" eigenvector). By ensuring that the range of interpolation exactly fits the approximate solution, as in the EIS, we obtain an interpolation operator that represents $\mathbf{x}$ to an arbitrarily high degree of accuracy as the approximate solution converges.

The rest of this chapter is organized as follows. We begin by describing the multiplicative correction framework for singular problems. In §3.3 we propose a modification of the classical AMG interpolation formula that results in a nonnegative interpolation operator with unit row sums. It is shown how a *lumping* technique (first proposed in [49]), maintains the sign structure and irreducibility of the coarse-level operators on all levels. The connection between the multiplicative correction framework and the standard multigrid additive correction framework is also discussed, which leads to a basic hybrid multiplicative/additive method. The chapter is concluded by numerical tests with a wide array of test problems for which traditional iterative methods are slow to converge.

In this chapter and all that follow we deviate from the multigrid terminology and two-level notation established in Chapter 2. In particular, the terms *grid* and *level* are used interchangeably. We note that for historical reasons *coarse-grid correction* remains unchanged. Coarse-level quantities are denoted by a subscript "c", while fine-level quantities and transfer operators do not carry any subscripts. When multilevel notation is necessary we proceed as in previous sections and append a subscript ℓ to indicate the level number, where $\ell = 0, \ldots, L$ for some positive integer L and zero indexes the finest level.

3.1 Multiplicative correction AMG framework

The primary difference between our approach for Markov chains and that of classical AMG for nonsingular linear systems (see §2.6.3) is the use of a multiplicative correction. Let $\mathbf{x}^{(k)}$ be the current approximation of the exact solution $\mathbf{x}$ prior to the pre-smoothing step, and let $\bar{\mathbf{x}}^{(k)}$ be the corresponding smoothed approximation of $\mathbf{x}$. Suppose there exist full rank restriction and interpolation operators, $\mathbf{R} \in \mathbb{R}^{n_c \times n}$ and $\mathbf{P} \in \mathbb{R}^{n \times n_c}$, respectively, such that $\bar{\mathbf{x}}^{(k)}$ is exactly in the range of $\mathbf{P}$. Further suppose that $\mathbf{x}$ is approximately in the range of $\mathbf{P}$, that is, $\mathbf{x} \approx \mathbf{P}\mathbf{x}_c$ for some coarse-level approximation $\mathbf{x}_c$. Clearly, the better $\bar{\mathbf{x}}^{(k)}$ approximates $\mathbf{x}$, the more accurately the exact solution is represented in the range of $\mathbf{P}$. The solution of the fine-level problem (3.2) can then be approximated by solving the coarse-level problem

$$\mathbf{A}_c\mathbf{x}_c = \mathbf{0}_c, \qquad \mu_c(\mathbf{x}_c) = 1, \tag{3.3}$$

where

$$\mathbf{A}_c := \mathbf{RAP} \tag{3.4}$$

is the $n_c \times n_c$ Galerkin coarse-level system operator. Because $\mathbf{A}$ has rank $n-1$ and the transfer operators are of full rank, the rank of $\mathbf{A}_c$ is $n_c - 1$, which implies that (3.3) has a unique solution. However, because $\mathbf{A}_c$ is generally not an irreducible singular M-matrix (owing to the structure of the transfer operators), the solution of the coarse-level problem may fail to be strictly positive. Hence, the positivity constraint has been dropped. The coarse-level normalization condition $\mu_c(\mathbf{x}_c) := \langle \mathbf{1}, \mathbf{P}\mathbf{x}_c\rangle$ ensures that the coarse-grid corrected approximation sums to one. We note that it is not actually necessary to impose the normalization condition on any of the coarse levels so long as the solution on the finest level is normalized after each iteration. After solving the coarse-level problem (3.3), the updated iterate is obtained by a *multiplicative* coarse-grid correction:

$$\mathbf{x}^{\text{CGC}} = \mathbf{P}\mathbf{x}_c. \tag{3.5}$$

The transfer operators are designed so that $\mathbf{A}_c$ accurately represents the left and right near-nullspace components of $\mathbf{A}$. Since $\mathbf{1}$ spans the left nullspace of $\mathbf{A}$, the restriction operator can be defined so that $\mathbf{1}$ is exactly represented in the range of $\mathbf{R}^\top$. Ideally, the interpolation contains the right nullspace component of $\mathbf{A}$ in its range. However, the right nullspace component $\mathbf{x}$ is not known (it is the target of our method), hence a fully accurate representation is not guaranteed. Instead, we compromise by ensuring that the relaxed iterate $\bar{\mathbf{x}}^{(k)}$, which should be *locally similar* to $\mathbf{x}$, lies exactly in the range of $\mathbf{P}$[1]. As the computed solution becomes more accurate, so too does the coarse representation of the original problem. To construct such operators, we begin by defining a full rank tentative interpolation operator $\tilde{\mathbf{P}}$. The sparsity structure of $\tilde{\mathbf{P}}$ is determined by the classical AMG two-pass coarsening with strength of connection based on the scaled operator

$$\bar{\mathbf{A}} = \mathbf{A}\mathrm{diag}(\bar{\mathbf{x}}^{(k)}) \tag{3.6}$$

(see Algorithms 2.3 and 2.4). The interpolation weights are computed by a variant of the classical AMG interpolation described in §3.3. In particular, $\tilde{\mathbf{P}}$ is guaranteed to have nonnegative elements and unit row sums, i.e., $\mathbf{1} = \tilde{\mathbf{P}}\mathbf{1}_c$. The restriction and interpolation operators are then defined by

$$\mathbf{R} := \tilde{\mathbf{P}}^\top \quad \text{and} \quad \mathbf{P} := \mathrm{diag}(\bar{\mathbf{x}}^{(k)})\tilde{\mathbf{P}}, \tag{3.7}$$

where $\mathbf{P}$ has full rank if and only if $\bar{\mathbf{x}}^{(k)}$ has nonzero entries. Consequently, interpolation is said to be *adaptive* because its range is updated after each iteration to exactly contain the most recent approximation of the exact solution. Since $\tilde{\mathbf{P}}$ interpolates the vector of all ones exactly,

$$\mathbf{1} \in \mathrm{range}(\mathbf{R}^\top) \quad \text{and} \quad \bar{\mathbf{x}}^{(k)} \in \mathrm{range}(\mathbf{P}). \tag{3.8}$$

[1]The vector $\bar{\mathbf{x}}^{(k)}$ is locally similar to $\mathbf{x}$ if about each point i there exists a small neighborhood $\mathcal{N}_i$ and a positive constant $c = c_i$ such that $\bar{x}_j^{(k)} \approx c x_j$ for all $j \in \mathcal{N}_i$. An assumption of AMG is that the near-nullspace components of $\mathbf{A}$ are locally similar, and hence any linear combination of them should be locally similar to $\mathbf{x}$. Consequently, saying $\bar{\mathbf{x}}^{(k)}$ is locally similar to $\mathbf{x}$ is equivalent to the statement that $\bar{\mathbf{x}}^{(k)}$ can be expressed as a linear combination of near-nullspace components of $\mathbf{A}$, including $\mathbf{x}$.

Moreover, if $\bar{\mathbf{x}}^{(k)} = \mathbf{x}$ then $\mathbf{x}_c = \mathbf{1}_c$ by the following argument

$$\mathbf{x} = \mathrm{diag}(\mathbf{x})\tilde{\mathbf{P}}\mathbf{x}_c \quad \Leftrightarrow \quad \mathbf{1} = \tilde{\mathbf{P}}\mathbf{x}_c \quad \Leftrightarrow \quad \mathbf{x}_c = \mathbf{1}_c. \tag{3.9}$$

To motivate our definition of the tentative interpolation operator $\tilde{\mathbf{P}}$, we define the *multiplicative error* of the relaxed fine-level approximation:

$$\mathbf{e} := \mathrm{diag}(\bar{\mathbf{x}}^{(k)})^{-1}\mathbf{x},$$

where it is assumed that $\bar{\mathbf{x}}^{(k)}$ has nonzero components. We note that if $\mathbf{x}^{(k)}$ is strictly positive on the finest level, and if the system operators are irreducible singular M-matrices on all levels, then it can be shown that the relaxed iterates have strictly positive components on all levels (see §3.5). The fine-level system can be formulated in terms of the multiplicative error as

$$\bar{\mathbf{A}}\mathbf{e} = \mathbf{0}, \tag{3.10}$$

and the coarse-grid correction, $\mathbf{P}\mathbf{x}_c$, can be interpreted as an approximation of the fine-level multiplicative error:

$$\mathbf{x} \approx \mathbf{P}\mathbf{x}_c \quad \Leftrightarrow \quad \mathbf{e} \approx \tilde{\mathbf{P}}\mathbf{x}_c. \tag{3.11}$$

Therefore, in order to obtain an effective interpolation the range of $\tilde{\mathbf{P}}$ must accurately represent multiplicative error corresponding to smoothed approximations $\bar{\mathbf{x}}^{(k)}$, which we refer to as algebraically smooth multiplicative error. To characterize algebraically smooth multiplicative error we note that relaxation on $\mathbf{A}\mathbf{x} = \mathbf{0}$ typically results in a small residual after just a few relaxation steps: $\mathbf{A}\bar{\mathbf{x}}^{(k)} \approx \mathbf{0}$. As the relaxed approximation $\bar{\mathbf{x}}^{(k)}$ approaches the exact solution, the multiplicative error $\mathbf{e}$ approaches $\mathbf{1}$. Moreover, (3.10) suggests that each component of $\mathbf{e}$ can be approximated by a weighted sum of neighboring error values, with weights proportional to the entries in $\bar{\mathbf{A}}$. These observations suggest that defining $\tilde{\mathbf{P}}$ via classical AMG operator interpolation with strength of connection based on $\bar{\mathbf{A}}$ should work well *assuming that smooth multiplicative error varies slowly along strong connections in* $\bar{\mathbf{A}}$. We make the following qualitative argument in support of this statement. Suppose that relaxation produces small residuals, that is, $\bar{\mathbf{A}}\mathbf{1} = \mathbf{A}\bar{\mathbf{x}}^{(k)} \approx \mathbf{0}$. Hence,

$$\sum_j \bar{a}_{ij} \approx 0 \quad \text{for each} \quad i. \tag{3.12}$$

By the multiplicative error equation

$$2\mathbf{e}^\top \bar{\mathbf{A}}\mathbf{e} = 0 \quad \Leftrightarrow \quad \sum_{ij} |\bar{a}_{ij}|(e_i - e_j)^2 = -\sum_{ij} \bar{a}_{ij} e_i^2 - \sum_{ij} \bar{a}_{ij} e_j^2. \tag{3.13}$$

Evaluating the terms on the right-hand side of this expression we find that

$$\begin{aligned} \sum_{ij} \bar{a}_{ij} e_i^2 &= \sum_i e_i^2 \sum_j \bar{a}_{ij} \approx 0 \quad \text{by (3.12)}, \\ \sum_{ij} \bar{a}_{ij} e_j^2 &= \sum_j e_j^2 \sum_i \bar{a}_{ij} = 0 \quad \text{since each column of } \bar{\mathbf{A}} \text{ sums to zero.} \end{aligned}$$

We note that in the above argument we assume that e_i^2 is bounded close to one, which is reasonable given that $\mathbf{e} \to \mathbf{1}$ as the $\bar{\mathbf{x}}^{(k)}$ converges to $\mathbf{x}$. Therefore,

$$\sum_i \sum_{j \neq i} |\bar{a}_{ij}|(e_i - e_j)^2 \approx 0. \tag{3.14}$$

This result implies that algebraically smooth multiplicative error varies slowly in the direction of relatively large off-diagonal elements of $\bar{\mathbf{A}}$. We conclude that for each i,

$$e_j \approx e_i \;\; \text{for all} \;\; j \in \mathcal{S}_i. \tag{3.15}$$

To further justify basing strength of connection on $\bar{\mathbf{A}}$ we offer the following probabilistic argument. The quantity $-a_{ij}$, for $j \neq i$, is the conditional probability of transitioning to state i given that the chain is currently in state j. Consider the last part of this statement. If the probability of being in state j is small (in the long run), then regardless of $-a_{ij}$, the probability of transitioning to state i from state j is also small. Therefore, we should consider the joint probability $-a_{ij}x_j$ to measure the influence of state j on state i. However, since $\mathbf{x}$ is unknown, we use the most recent approximation of the exact solution and base strength of connection on $\mathbf{A}\,\mathrm{diag}(\bar{\mathbf{x}}^{(k)})$. The system operator on the coarse level, $\hat{\mathbf{A}}_c$, is also an irreducible singular M-matrix with zero column sums that is obtained from $\mathbf{A}_c$ through a lumping process (see §3.4). Therefore, by Definition 2.2.2,

$$\hat{\mathbf{A}}_c = \rho(\mathbf{B}_c)\mathbf{I}_c - \mathbf{B}_c,$$

where $\mathbf{B}_c$ is some irreducible nonnegative matrix and $\rho(\mathbf{B}_c) > 0$ is the spectral radius of $\mathbf{B}_c$. Letting $\alpha = 1/\rho(\mathbf{B}_c)$, we have that

$$\alpha\hat{\mathbf{A}}_c = \mathbf{I}_c - \alpha\mathbf{B}_c,$$

where the scaled matrix $\alpha\mathbf{B}_c$ is an irreducible column-stochastic matrix. Hence, $\hat{\mathbf{A}}_c$ corresponds to a coarse-level Markov chain with transition matrix $\mathbf{B}_c$.

Remark 3.1.1. In practice we scale $\bar{\mathbf{A}}$ by the reciprocal of its maximum diagonal element. Doing so helps to mitigate the occurrence of extremely small off-diagonal elements in $\mathbf{A}_c$, which if present may negatively affect the lumping routine described below due to numerical roundoff error. We note that the coarsening routine and the formula for the interpolation weights are invariant to such scaling.

Due to the structure of the transfer operators the coarse-level system operator, $\mathbf{A}_c$, may not be an irreducible singular M-matrix; hence, the solution of the coarse-level problem may not be strictly positive. In this case there is no way to guarantee strict positivity of the iterates on all levels. To remedy this issue we apply a *lumping method* (see §3.4) that computes a *lumped* coarse-level system operator $\hat{\mathbf{A}}_c$, which is an irreducible singular M-matrix, by adding small perturbations to some of the elements of $\mathbf{A}_c$. Instead of solving (3.3), we solve the lumped coarse-level problem:

$$\hat{\mathbf{A}}_c\mathbf{x}_c = \mathbf{0}_c, \qquad (\mathbf{x}_c)_i > 0 \ \text{ for } \ i = 1, \dots, n_c, \qquad \mu_c(\mathbf{x}_c) = 1. \tag{3.16}$$

We note that the positivity constraint on $\mathbf{x}_c$ is now justified. In §3.5 we prove that lumping leads to irreducible singular M-matrices on all levels, which implies the strict positivity of the iterates on all levels. Moreover, we prove that the exact solution is a fixed point of the multilevel V-cycle with the lumped coarse-level operators. In order to prove these claims we require that

$$\mathbf{1}_c^\top \mathbf{A}_c = \mathbf{0}_c \quad \text{for all iterates } \bar{\mathbf{x}}^{(k)}, \tag{3.17}$$

$$\mathbf{A}_c \mathbf{1}_c = \mathbf{0}_c \quad \text{for } \bar{\mathbf{x}}^{(k)} = \mathbf{x}. \tag{3.18}$$

Given that $\mathbf{A}$ is an irreducible singular M-matrix with zero column sums, conditions (3.17) and (3.18) are clearly satisfied by the properties of the transfer operators (3.8).

Algorithm 3.1: MCAMG V-cycle for Markov chains

```
   Input: $\mathbf{A}_\ell$, current approximation $\mathbf{x}_\ell^{(k)}$, number of smoothing steps $\nu_1$, $\nu_2$
   Output: New approximation $\mathbf{x}_\ell^{(k+1)}$
   if on the coarsest level then
1.     Solve $\mathbf{A}_\ell \mathbf{x}_\ell = \mathbf{0}_\ell$ subject to $\mathbf{x}_\ell$ strictly positive
   else
2.     Perform $\nu_1$ relaxations: $\bar{\mathbf{x}}_\ell^{(k)} \leftarrow \mathrm{Relax}(\mathbf{A}_\ell, \mathbf{0}_\ell, \mathbf{x}_\ell^{(k)}, \nu_1)$
3.     Compute the transfer operators $\mathbf{R} \leftarrow \tilde{\mathbf{P}}^\top$ and $\mathbf{P} \leftarrow \mathrm{diag}(\bar{\mathbf{x}}_\ell^{(k)})\tilde{\mathbf{P}}$
4.     Construct the coarse-level system operator $\mathbf{A}_{\ell+1} \leftarrow \mathbf{R}\mathbf{A}_\ell\mathbf{P}$
5.     Compute the lumped coarse-level system operator $\hat{\mathbf{A}}_{\ell+1}$
6.     Recursive call:
               $\mathbf{x}_{\ell+1} \leftarrow \mathrm{MCAMG}(\hat{\mathbf{A}}_{\ell+1}, \mathbf{1}_{\ell+1}, \nu_1, \nu_2)$
7.     Correct: $\mathbf{x}_\ell^{\mathrm{CGC}} \leftarrow \mathbf{P}\mathbf{x}_{\ell+1}$
8.     Perform $\nu_2$ relaxations: $\mathbf{x}_\ell^{(k+1)} \leftarrow \mathrm{Relax}(\mathbf{A}_\ell, \mathbf{0}_\ell, \mathbf{x}_\ell^{\mathrm{CGC}}, \nu_2)$
   end
```

We conclude this section by stating the multilevel V-cycle MCAMG (Markov chain AMG) method given by Algorithm 3.1. As our smoother we use the weighted Jacobi method with relaxation parameter $\omega \in (0, 1)$. The coarsest level is reached when the number of points on the current level is less than some threshold value n_{coarse}. The maximum number of levels can also be restricted to some positive integer L if, for example, we want a two- or three-level solver. As our coarsest-level solver we use a variant of Gaussian elimination which is briefly described in the next section. We note that since $\mathbf{A}_\ell$ depends on $\bar{\mathbf{x}}_{\ell-1}^{(k)}$ for $\ell = 1, 2, \ldots, L$, in principle it is necessary to recompute the transfer operators on all levels in each V-cycle. To reduce the amount of work per cycle, the tentative interpolation operator can be frozen on all levels after only a few iterations; however, it remains necessary to recompute the coarse-level system operators each cycle according to (3.4) and (3.7) so that conditions (3.9) and (3.18) are satisfied. Because the computed solution is locally similar to the exact solution after only a few multigrid cycles, freezing the tentative interpolation typically does not lead to a deterioration in the rate of convergence.

3.2 Coarsest-level solver: GTH algorithm

According to Theorem 3.5.1 the system operator on the coarsest level is an irreducible singular M-matrix. Consequently, we may use the Grassmann–Taksar–Heyman (GTH) algorithm [61] as the direct solver on the coarsest level. The GTH algorithm is a variant of Gaussian elimination for computing the stationary distribution of an irreducible Markov chain, or equivalently, the unique nullspace vector of an irreducible singular M-matrix with zero column sums. The key to the GTH algorithm is that after each reduction step (zeroing the sub-diagonal elements in a column) the remaining unreduced portion of the matrix is itself a singular M-matrix with zero column sums that corresponds to a Markov chain defined on a reduced set of states. Consequently, diagonal elements can be computed as the negated sum the off-diagonal elements, which avoids any subtractions. Since diagonal elements computed during the elimination process are guaranteed to be nonzero, pivoting is unnecessary, and cancellation by loss of significant digits is avoided, which improves stability. In fact, the GTH algorithm is forward stable and is guaranteed to compute the stationary vector with low relative error in each component. For implementation details regarding the GTH algorithm we refer to [110, 111].

3.3 Interpolation

The sparsity structure of the tentative interpolation operator $\tilde{\mathbf{P}}$ is defined according to the definition of the classical AMG interpolation operator presented in §2.6.3. That is, each C-point interpolates from its corresponding coarse-level point, and the interpolated value of each F-point i is given by a weighted sum of the values of the points in its coarse interpolatory set $\mathcal{C}_i$ (see (2.58) for details). The interpolation weights $\tilde{w}_{ij}$ are defined so that $\tilde{\mathbf{P}}$ has nonnegative elements and unit row sums. For any F-point i we have that

$$(\tilde{\mathbf{P}})_{i,\alpha(j)} = \tilde{w}_{ij} = \frac{\bar{a}_{ij} + \sum_{m\in\mathcal{D}_i^s}\left(\frac{\bar{a}_{im}\bar{a}_{mj}}{\sum_{k\in\mathcal{C}_i}\bar{a}_{mk}}\right)}{\sum_{p\in\mathcal{C}_i}\bar{a}_{ip} + \sum_{r\in\mathcal{D}_i^s}\bar{a}_{ir}} \quad \text{for all} \quad j \in \mathcal{C}_i, \tag{3.19}$$

where $\alpha(j)$ is the coarse-level label for C-point j, and $\mathcal{D}_i^s$ is the set of F-points that strongly influence point i. Comparison with the classical AMG definition of interpolation shows that (3.19) is essentially a rescaled version of the original formula (2.61). This rescaling is necessary because the classical interpolation formula applied to singular M-matrices can lead to negative weights, and even division by zero in the case that $\mathbf{A}$ is not row diagonally dominant. It is, however, easy to verify that the rescaled formula does not suffer from these deficiencies, as we now explain.

Under the premise that $\mathbf{A}$ is an irreducible singular M-matrix, and by the fact that $i \notin \mathcal{C}_i \cup \mathcal{D}_i^s$, it follows that all matrix elements used in (3.19) are nonpositive. The two-pass AMG coarsening routine ensures that $\mathcal{C}_i \neq \emptyset$, hence the denominator in (3.19) is nonzero. Together with the fact that $\mathcal{C}_i \cap \mathcal{D}_i^s = \emptyset$, which precludes diagonal elements $\bar{a}_{mm}$ from occurring in (3.19), we find that $\tilde{w}_{ij} > 0$ for all $i \in \mathcal{F}$ and $j \in \mathcal{C}_i$. Thus, $\tilde{\mathbf{P}}$ has nonnegative entries. By computing the sum $\sum_{j\in\mathcal{C}_i}\tilde{w}_{ij}$, we observe that $\tilde{\mathbf{P}}$ has unit row sums. We note that it is important to perform both passes of the coarsening routine, since this ensures that $\sum_{k\in\mathcal{C}_i}\bar{a}_{mk} \neq 0$ for any $i \in \mathcal{F}$ and $m \in \mathcal{D}_i^s$, which is required for the $\tilde{w}_{ij}$s to be well-defined. In particular, the second pass of the coarsening routine ensures that every point in $\mathcal{D}_i^s$ strongly depends on at least one point in $\mathcal{C}_i$.

We conclude this section by mentioning an alternative formulation of the interpolation weights that was suggested to us by multigrid guru Steve McCormick. Essentially, instead of collapsing the weak connections in $\mathcal{D}_i^w$ to the diagonal they are included with the set of strong connections $\mathcal{D}_i^s$. Following the derivation of the classical AMG interpolation weights formula in §2.6.3, and normalizing the weights to sum to one, we find that

$$w_{ij}^{\text{SM}} = \frac{\bar{a}_{ij} + \sum_{m \in \mathcal{D}_i^s \cap \mathcal{D}_i^w} \left(\frac{\bar{a}_{im}\bar{a}_{mj}}{\sum_{k \in \mathcal{C}_i} \bar{a}_{mk}} \right)}{\sum_{p \in \mathcal{N}_i} \bar{a}_{ip}} > 0 \quad \text{for all} \quad j \in \mathcal{C}_i. \tag{3.20}$$

Because weak connections in $\mathcal{D}_i^w$ correspond to small elements in row i of $\bar{\mathbf{A}}$ (relative to those elements in row i that correspond to strong connections), it should be true that

$$w_{ij}^{\text{SM}} \approx \tilde{w}_{ij} + \frac{1}{\sum_{p \in \mathcal{N}_i} \bar{a}_{ip}} \sum_{m \in \mathcal{D}_i^w} \left(\frac{\bar{a}_{im}\bar{a}_{mj}}{\sum_{k \in \mathcal{C}_i} \bar{a}_{mk}} \right) \quad \text{for all} \quad j \in \mathcal{C}_i. \tag{3.21}$$

The summation on the right-hand side of this expression is over $\mathcal{D}_i^w$, hence its value should be close to zero. Therefore, the interpolation formula suggested by Steve McCormick gives approximately the same interpolation weights as our formula. In practice, either of these formulas result in very satisfactory and comparable performance of our multigrid method. In this thesis, and in general, we use formula (3.19) to define the interpolation weights.

3.4 Lumping

Owing to the structure of $\tilde{\mathbf{P}}$, the coarse-level operator $\mathbf{A}_c$ may not be an irreducible singular M-matrix. Let $\bar{\mathbf{A}} = \mathbf{A}\text{diag}(\bar{\mathbf{x}}^{(k)})$ and consider the splitting $\bar{\mathbf{A}} = \mathbf{D} - (\mathbf{L} + \mathbf{U})$, where $\mathbf{D} = \text{diag}(\bar{\mathbf{A}})$, $\mathbf{L}$ is strictly lower triangular, and $\mathbf{U}$ is strictly upper triangular. Then

$$\mathbf{A}_c = \tilde{\mathbf{P}}^\top \bar{\mathbf{A}} \tilde{\mathbf{P}} = \tilde{\mathbf{P}}^\top \mathbf{D} \tilde{\mathbf{P}} - \tilde{\mathbf{P}}^\top (\mathbf{L} + \mathbf{U}) \tilde{\mathbf{P}} = \mathbf{S}_c - \mathbf{G}_c, \tag{3.22}$$

where $\mathbf{S}_c = \tilde{\mathbf{P}}^\top \mathbf{D} \tilde{\mathbf{P}}$ and $\mathbf{G}_c = \tilde{\mathbf{P}}^\top (\mathbf{L}+\mathbf{U}) \tilde{\mathbf{P}}$ are nonnegative matrices because $\bar{\mathbf{A}}$ is a singular M-matrix and $\tilde{\mathbf{P}}$ has nonnegative elements. Since $\mathbf{S}_c$ is generally not diagonal, $\mathbf{A}_c$ may have positive off-diagonal elements, and therefore may not be a singular M-matrix. Furthermore,

$\mathbf{A}_c$ may lose irreducibility due to the creation of new zero elements. We address this problem by applying the lumping method described in [49]. In essence, a modified coarse-level system operator is constructed by symmetrically *lumping* off-diagonal weight in $\mathbf{S}_c$ to the diagonal. The resulting lumped operator $\hat{\mathbf{A}}_c$ has nonpositive off-diagonal elements and retains nonzero off-diagonal elements where $\mathbf{G}_c$ has nonzero off-diagonal elements (to guarantee irreducibility). It is important to note that while lumping may introduce new nonzero entries into $\hat{\mathbf{A}}_c$, it cannot create a zero entry in $\hat{\mathbf{A}}_c$ where $\mathbf{G}_c$ is nonzero.

Without lumping our method often displays erratic convergence behavior, in some cases stalling or even diverging. There are many potential pitfalls when the coarse-level operators are not irreducible singular M-matrices. For example, incorrect signs in coarse-level operators may result in negative interpolation weights. Coarse-grid correction may then lead to the generation of vectors with vanishing or negative components. Moreover, components with incorrect signs may be generated after relaxation. These pathological vectors propagate incorrect signs upward in the cycle via coarse-grid correction, and downward via column-scaled operators that may have entire columns that vanish or have incorrect signs. A loss of irreducibility means that a strictly positive solution is no longer guaranteed for the direct solve on the coarsest level. In certain cases we have found that lumping is unnecessary; however, we do not know of an easy way to determine a priori if a particular problem requires lumping. In our experience, matrices that are similar to symmetric matrices, and thus have real eigenvalue spectra, often do not require lumping, except perhaps in the first few cycles. However, even in such idealized cases, if lumping is not performed in the early cycles then convergence may become erratic, especially for large problems. In general, problems with less symmetry typically require lumping in all cycles. Consequently, lumping is paramount to the robustness of our algorithm.

In order to describe the lumping procedure in more detail we define an *offending index pair*, which is an ordered pair (i, j) such that $i \neq j$, $s_{ij} \neq 0$, and $(\mathbf{A}_c)_{ij} \geq 0$. It is precisely for these indices that lumping is performed. Let (i, j) be an offending index pair. To correct the sign in $\mathbf{A}_c$ at the ijth location we define the matrix

$$\mathbf{S}_{\{i,j\}} := \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \left[\begin{array}{ccccc} \ddots & \overset{i}{\vdots} & & \overset{j}{\vdots} & \\ \cdots & \beta_{\{i,j\}} & \cdots & -\beta_{\{i,j\}} & \cdots \\ & \vdots & & \vdots & \\ \cdots & -\beta_{\{i,j\}} & \cdots & \beta_{\{i,j\}} & \cdots \\ & \vdots & & \vdots & \end{array} \right], \tag{3.23}$$

where $\beta_{\{i,j\}} > 0$, and the other elements are zero. Adding $\mathbf{S}_{\{i,j\}}$ to $\mathbf{S}_c = (s_{ij})$ corresponds to lumping parts of $\mathbf{S}_c$ to the diagonal, in the sense that $\beta_{\{i,j\}}$ is removed from off-diagonal elements s_{ij} and s_{ji} and added to diagonal elements s_{ii} and s_{jj}. By choosing $\beta_{\{i,j\}}$ so that the following inequalities are satisfied,

$$s_{ij} - g_{ij} - \beta_{\{i,j\}} < 0,$$
$$s_{ji} - g_{ji} - \beta_{\{i,j\}} < 0,$$

we ensure that the lumped operator $\hat{\mathbf{A}}_c$ has strictly negative elements at the ijth and jith locations. We note that the lumping procedure is symmetric in the sense that correcting the sign at the ijth location also corrects the sign at the jith location (if necessary). Hence, if (i, j) and (j, i) are offending index pairs then only one matrix $\mathbf{S}_{\{i,j\}}$ must be added to $\mathbf{S}_c$. Moreover, the symmetry of $\mathbf{S}_{\{i,j\}}$ preserves the column sums and row sums of $\mathbf{A}_c$, which ensures that $\hat{\mathbf{A}}_c$ satisfies conditions (3.17) and (3.18). Indeed, if $\tilde{\mathbf{S}}_c = \sum \mathbf{S}_{\{i,j\}}$, where the sum is over the lumped index pairs, then

$$\mathbf{1}_c^\top \hat{\mathbf{A}}_c = \mathbf{1}_c^\top \mathbf{A}_c + \mathbf{1}_c^\top \tilde{\mathbf{S}}_c = \mathbf{1}_c^\top \mathbf{A}_c = \mathbf{0} \quad \text{for all iterates } \bar{\mathbf{x}}^{(k)}, \tag{3.24}$$

$$\mathbf{A}_c \mathbf{1}_c = \mathbf{A}_c \mathbf{1}_c + \tilde{\mathbf{S}}_c \mathbf{1}_c = \mathbf{A}_c \mathbf{1}_c = \mathbf{0} \quad \text{for } \bar{\mathbf{x}}^{(k)} = \mathbf{x}. \tag{3.25}$$

In our implementation $\beta_{\{i,j\}}$ is the maximum of $\beta^{(1)}_{\{i,j\}}$ and $\beta^{(2)}_{\{i,j\}}$, where

$$s_{ij} - g_{ij} - \beta^{(1)}_{\{i,j\}} = -\eta\, g_{ij},$$
$$s_{ji} - g_{ji} - \beta^{(2)}_{\{i,j\}} = -\eta\, g_{ji},$$

and $\eta \in (0, 1]$ is a fixed parameter. Empirical evidence suggests that we should lump as little as possible [49], and in practice $\eta = 0.01$ seems to work well. We note that since $\mathbf{S}_c$ remains a symmetric matrix throughout the lumping process, it is sufficient to examine its strictly lower (or strictly upper) triangular part for detecting potential offending index pairs.

3.5 Fixed point property of MCAMG

In this section we prove that the exact solution is a fixed point of the MCAMG V-cycle. In what follows we let $\ell = 0, \ldots, L$ index the level of a V-cycle, and we let $\mathbf{P}^{\ell}_{\ell+1}$ and $\mathbf{R}^{\ell+1}_{\ell}$ denote the interpolation and restriction operators between levels ℓ and $\ell+1$, respectively. We begin by proving that the coarse-level matrix $\mathbf{G}_c$ in (3.22) is irreducible if $\bar{\mathbf{A}}$ is irreducible.

Proposition 3.5.1 (Irreducibility of $\mathbf{G}_c$)**.** *If $\bar{\mathbf{A}} = \mathbf{D} - (\mathbf{L} + \mathbf{U})$ is an irreducible singular M-matrix then $\mathbf{G}_c = \tilde{\mathbf{P}}^{\top}(\mathbf{L} + \mathbf{U})\tilde{\mathbf{P}}$ is irreducible.*

Proof. It is sufficient to show that for any C-points with coarse-level labels $I \neq J$, there exists a directed path from node I to node J in the directed graph of $\mathbf{G}_c = (g_{ij})$. First, observe that irreducibility of $\bar{\mathbf{A}}$ implies irreducibility of $\mathbf{L} + \mathbf{U}$. Let $i \neq j$ be any fine-level labels such that $(\mathbf{L} + \mathbf{U})_{ij} \neq 0$. Furthermore, let I be any C-point that interpolates to i, i.e., $\tilde{p}_{iI} \neq 0$, and let J be any C-point that interpolates to j, i.e., $\tilde{p}_{jJ} \neq 0$. Since every row of $\tilde{\mathbf{P}}$ contains at least one nonzero element (its rows sum to one), indices I and J exist. Now,

$$g_{IJ} = \sum_{m,n} \tilde{p}_{mI}(l_{mn} + u_{mn})\tilde{p}_{nJ},$$

and because $\tilde{p}_{iI}$, $\tilde{p}_{jJ}$, and $(\mathbf{L}+\mathbf{U})_{ij}$ are nonzero, it follows $g_{IJ} \neq 0$. Thus, for any fine-level points i and j such that an arc exists from node i to node j in $\mathcal{D}(\bar{\mathbf{A}})$, there exist coarse-level points I and J such that an arc exists from node J to node I in $\mathcal{D}(\mathbf{G}_c)$.

Now, let I and J be any two distinct C-points. Furthermore, let i and j be the fine-level labels of I and J, respectively. By irreducibility of $\mathbf{L}+\mathbf{U}$ there exists a directed path of distinct fine-level points from node i to node j. Denote this path by

$$i = v_0, v_1, \ldots, v_{k-1}, v_k = j,$$

where nodes $v_0, \ldots, v_k$ are fine-level points. By the result above there must exist coarse-level points $V_0, \ldots, V_k$ that form a directed walk (see §2.2)

$$V_0, V_1, \ldots, V_{k-1}, V_k$$

in $\mathcal{D}(\mathbf{G}_c)$. However, any directed U-V walk contains a directed U-V path [36]. Thus, there exists a directed path in $\mathcal{D}(\mathbf{G}_c)$ that begins at V_0 and ends at V_k. Recall that C-points V_0 and V_k were chosen such that they interpolate to $v_0 = i$ and $v_k = j$, respectively. Since the only point that interpolates to a given C-point is the point itself (by the definition of interpolation), it follows that $V_0 = I$ and $V_k = J$. Hence, there exists a directed path from node I to node J in the directed graph of $\mathbf{G}_c$. Since I and J were arbitrary, $\mathbf{G}_c$ is irreducible. $\square$

Theorem 3.5.1 (Singular M-matrix property of an MCAMG V-cycle)**.** *Assume that* $\mathbf{A}_0$ *is an irreducible singular M-matrix and that* $\bar{\mathbf{x}}_0^{(k)}$ *has strictly positive components. Then the coarse-level operators* $\mathbf{A}_1, \ldots, \mathbf{A}_L$ *corresponding to an MCAMG V-cycle are irreducible singular M-matrices, and the pre-smoothed iterates* $\bar{\mathbf{x}}_1^{(k)}, \ldots, \bar{\mathbf{x}}_{L-1}^{(k)}$ *have strictly positive components.*

Proof. The proof is by induction. Since $\bar{\mathbf{x}}_0^{(k)}$ is strictly positive the scaled operator $\bar{\mathbf{A}}_0 = \mathbf{A}_0 \mathrm{diag}(\bar{\mathbf{x}}_0^{(k)})$ is an irreducible singular M-matrix. Consider the splitting $\bar{\mathbf{A}}_0 = \mathbf{D}_0 - (\mathbf{D}_0 -$

$\bar{\mathbf{A}}_0$) with $\mathbf{D}_0 = \operatorname{diag}(\bar{\mathbf{A}}_0)$. By Proposition 3.5.1 the matrix

$$\mathbf{G}_1 = (\tilde{\mathbf{P}}_1^0)^\top (\mathbf{D}_0 - \bar{\mathbf{A}}_0)\tilde{\mathbf{P}}_1^0$$

is irreducible. Lumping ensures that $\mathbf{A}_1$ has nonpositive off-diagonal elements and the same nonzero sparsity pattern as $\mathbf{G}_1$. Therefore, $\mathbf{A}_1$ is irreducible, and by (3.24) in conjunction with Theorem 2.2.6 (2) it follows that $\mathbf{A}_1$ is a singular M-matrix. Strict positivity of $\bar{\mathbf{x}}_1^{(k)}$ follows by Theorem 2.5.1 and by the fact that $\mathbf{x}_1^{(k)} = \mathbf{1}_1$.

Now suppose that for some $\ell \geq 0$ the matrix $\mathbf{A}_\ell$ is an irreducible singular M-matrix and the iterate $\bar{\mathbf{x}}_\ell^{(k)}$ is strictly positive. Then the scaled operator $\bar{\mathbf{A}}_\ell$ is an irreducible singular M-matrix. Hence, by Proposition 3.5.1 the matrix

$$\mathbf{G}_{\ell+1} = (\tilde{\mathbf{P}}_{\ell+1}^\ell)^\top (\operatorname{diag}(\bar{\mathbf{A}}_{\ell+1}) - \bar{\mathbf{A}}_{\ell+1})\tilde{\mathbf{P}}_{\ell+1}^\ell$$

is irreducible. Lumping ensures that $\mathbf{A}_{\ell+1}$ has nonpositive off-diagonal elements and same nonzero sparsity pattern as $\mathbf{G}_{\ell+1}$. Therefore, $\mathbf{A}_{\ell+1}$ is irreducible, and by (3.24) in conjunction with Theorem 2.2.6 (2) it follows that $\mathbf{A}_{\ell+1}$ is a singular M-matrix. Strict positivity of $\bar{\mathbf{x}}_{\ell+1}^{(k)}$ follows by Theorem 2.5.1 and by the fact that $\mathbf{x}_{\ell+1}^{(k)} = \mathbf{1}_{\ell+1}$. □

Corollary 3.5.1. *The system operators $\mathbf{A}_0, \ldots, \mathbf{A}_L$ corresponding to an MCAMG V-cycle each have a unique right nullspace vector with positive components (up to scaling).*

Proof. Since $\mathbf{A}_\ell$ is an irreducible singular M-matrix for $\ell = 0, \ldots, L$, it follows by Theorem 2.2.6 (1) that $\mathbf{A}_\ell$ has a unique right nullspace vector with positive components (up to scaling). □

Corollary 3.5.2. *If $\mathbf{A}_0$ is an irreducible singular M-matrix, and if $\bar{\mathbf{x}}_0^{(k)}$ is strictly positive, then the relaxed coarse-grid corrections produced by an MCAMG V-cycle have strictly positive components on all levels.*

Proof. By Theorem 3.5.1 the pre-smoothed iterates, $\bar{\mathbf{x}}_1^{(k)}, \ldots, \bar{\mathbf{x}}_{L-1}^{(k)}$, have strictly positive components. Moreover, the solution of the coarsest-level problem, $\mathbf{A}_L\mathbf{x}_L = \mathbf{0}_L$, is strictly positive because $\mathbf{A}_L$ is an irreducible singular M-matrix. Since the tentative interpolation operators, $\tilde{\mathbf{P}}^{\ell}_{\ell+1}$, are nonnegative with at least one nonzero element per row, it follows that

$$\mathbf{x}^{\text{CGC}}_{L-1} = \text{diag}(\bar{\mathbf{x}}^{(k)}_{L-1})\tilde{\mathbf{P}}^{L-1}_{L}\mathbf{x}_L$$

is strictly positive. Thus by Theorem 2.5.6, the post-smoothed coarse-grid corrected iterate, $\mathbf{x}^{(k+1)}_{L-1}$, is strictly positive. Continuing this argument inductively up to the finest level it follows that $\mathbf{x}^{(k+1)}_{\ell}$ is strictly positive for $\ell = L-1, \ldots, 0$. $\square$

An immediate result of corollary 3.5.2 is that the interpolation operators must have full rank on all levels. We conclude this section by proving a fixed point theorem for the MCAMG V-cycle algorithm. In what follows we let

$$\mathbf{H}^{\nu}_{\ell} = (\mathbf{I}_{\ell} - \mathbf{M}^{-1}_{\ell}\mathbf{A}_{\ell})^{\nu}$$

denote ν iterations of a general relaxation scheme on level ℓ.

Lemma 3.5.1. *If the current iterate on the finest level of an MCAMG V-cycle is equal to the exact solution then the relaxed solution $\bar{\mathbf{x}}^{(k)}_{\ell} = \mathbf{1}_{\ell}$ for $\ell = 1, \ldots, L-1$. Moreover, $\mathbf{A}_{\ell}\mathbf{1}_{\ell} = \mathbf{0}_{\ell}$ for $\ell = 1, \ldots, L$.*

Proof. The proof is by induction. Suppose the current iterate $\mathbf{x}^{(k)}_0$ is equal to the exact solution $\mathbf{x}$ on the finest level. Then

$$\mathbf{A}_1 = \mathbf{R}^1_0\,\mathbf{A}\,\text{diag}(\mathbf{x})\tilde{\mathbf{P}}^0_1 + \tilde{\mathbf{S}}_1,$$

where $\tilde{\mathbf{S}}_1$ is the lumping correction. Since the tentative interpolation operators have unit row sums on all levels, and since the lumping matrices have zero row sums on all levels,

$$\mathbf{A}_1\mathbf{1}_1 = \mathbf{R}^1_0\,\mathbf{A}\mathbf{x} = \mathbf{0}_1.$$

It immediately follows that $\bar{\mathbf{x}}^{(k)}_1 = \mathbf{H}^{\nu}_1\mathbf{1}_1 = \mathbf{1}_1$ for any number of iterations $\nu \geq 0$. Now suppose the induction hypothesis is true for some level $\ell \geq 1$, so $\mathbf{x}_\ell^{(k)} = \mathbf{1}_\ell$. Then

$$\mathbf{A}_{\ell+1} = \mathbf{R}_\ell^{\ell+1} \mathbf{A}_\ell \operatorname{diag}(\mathbf{1}_\ell) \tilde{\mathbf{P}}_{\ell+1}^\ell + \tilde{\mathbf{S}}_{\ell+1},$$

and by the induction hypothesis

$$\mathbf{A}_{\ell+1} \mathbf{1}_\ell = \mathbf{R}_k^{\ell+1} \mathbf{A}_\ell \mathbf{1}_\ell = \mathbf{0}_\ell.$$

It follows immediately that $\bar{\mathbf{x}}_\ell^{(k)} = \mathbf{H}_{\ell+1}^\nu \mathbf{1}_{\ell+1} = \mathbf{1}_{\ell+1}$ for any number of iterations $\nu \geq 0$. □

Theorem 3.5.2 (MCAMG V-cycle fixed point property)**.** *The exact solution* $\mathbf{x}$ *is a fixed point of the MCAMG V-cycle method.*

Proof. If $\mathbf{x}_0^{(k)} = \mathbf{x}$ then by Lemma 3.5.1 the solution of coarsest-level problem is $\mathbf{x}_L = \mathbf{1}_L$. Since $\bar{\mathbf{x}}_\ell^{(k)} = \mathbf{1}_\ell$ for $\ell = 1, \ldots, L-1$, on the finest level it follows that

$$\begin{aligned}
\mathbf{x}_0^{\mathrm{CGC}} &= \mathbf{H}_0^{\nu_2} \operatorname{diag}(\mathbf{x}) \tilde{\mathbf{P}}_1^0 \, \mathbf{H}_1^{\nu_2} \tilde{\mathbf{P}}_2^1 \ldots \mathbf{H}_{L-1}^{\nu_2} \tilde{\mathbf{P}}_L^{L-1} \mathbf{1}_L \\
&= \mathbf{H}_0^{\nu_2} \operatorname{diag}(\mathbf{x}) \tilde{\mathbf{P}}_1^0 \mathbf{1}_1 \\
&= \mathbf{H}_0^{\nu_2} \mathbf{x} \\
&= \mathbf{x}.
\end{aligned}$$

The conclusion now follows by the fact that the exact solution is a fixed point of the relaxation scheme on the finest level. □

3.6 Multiplicative correction vs. additive correction

We begin this section by showing that under certain assumptions the standard multigrid additive correction scheme is equivalent to the multiplicative correction scheme, without lumping, presented in §3.1. We then define a hybrid method that combines the MCAMG method (with lumping) as a setup phase in conjunction with a standard multigrid additive correction scheme as a solve phase. We begin by describing the standard additive multigrid correction scheme for the problem $\mathbf{Ax} = \mathbf{0}$. Given the current relaxed fine-level approximation $\bar{\mathbf{x}}^{(k)}$ (the same as in the multiplicative scheme), the additive error is defined by

$$\mathbf{e}^{add} := \mathbf{x} - \bar{\mathbf{x}}^{(k)}. \tag{3.26}$$

The fine-level problem can be rewritten in terms of the additive error as the fine-level residual equation

$$\mathbf{A}\mathbf{e}^{add} = \mathbf{r}^{(k)} = -\mathbf{A}\bar{\mathbf{x}}^{(k)}. \tag{3.27}$$

We seek a coarse-level approximation of the error $\mathbf{e}_c^{add}$, that when interpolated to the fine level approximately equals the unknown fine-level additive error. The coarse-level error is obtained by solving (approximately) the coarse-level problem

$$\mathbf{RAP}\mathbf{e}_c^{add} = \mathbf{R}\mathbf{r}^{(k)}, \tag{3.28}$$

where $\mathbf{R}$ is the $n_c \times n$ full rank restriction operator, and $\mathbf{P}$ is the $n \times n_c$ full rank interpolation operator. The updated fine-level approximation is then given by the coarse-grid correction

$$\mathbf{x}^{\text{CGC}} = \bar{\mathbf{x}}^{(k)} + \mathbf{P}\mathbf{e}_c^{add}. \tag{3.29}$$

We recall that in order for this process to be effective the smooth additive error components must lie approximately in the range of interpolation (so they can be removed by coarse-grid correction). Now consider the interpolation operator $\mathbf{P} = \text{diag}(\bar{\mathbf{x}}^{(k)})\tilde{\mathbf{P}}$ for the multiplicative correction scheme. The near-nullspace components of $\mathbf{A}$, that is, the components of the algebraically smooth additive error $\mathbf{e}^{add}$, are locally similar to $\mathbf{x}$ and $\bar{\mathbf{x}}^{(k)}$. Since $\bar{\mathbf{x}}^{(k)}$ is locally similar to $\mathbf{x}$, and $\bar{\mathbf{x}}^{(k)} \in \text{range}(\mathbf{P})$, the additive error should also lie approximately in the range of $\mathbf{P}$. Therefore, the multiplicative correction scheme interpolation should also be a suitable interpolation for the additive scheme. In fact, if the (adaptive) interpolation and restriction for the multiplicative scheme are used in every iteration of the additive scheme, then the multiplicative and additive schemes are equivalent.

Proposition 3.6.1 (Equivalence of multiplicative and additive schemes)**.** *One two-level MCAMG cycle without lumping is equivalent to one iteration of the additive scheme, assuming the same initial guess, the same transfer operators, and the same number of pre- and post-relaxations are used in both cycles.*

Proof. Let $\mathbf{R}$ and $\mathbf{P} = \text{diag}(\bar{\mathbf{x}}^{(k)})\tilde{\mathbf{P}}$ be the transfer operators from the multiplicative cycle. Since $\bar{\mathbf{x}}^{(k)}$ is in the range of $\mathbf{P}$, the coarse-level problem for the additive scheme is equivalent to

$$\mathbf{RAPe}_c^{add} = \mathbf{Rr}^{(k)} = -\mathbf{RAP1}_c \quad \Leftrightarrow \quad \mathbf{RAP}(\mathbf{e}_c^{add} + \mathbf{1}_c) = \mathbf{0}_c.$$

Letting $\mathbf{x}_c = \mathbf{e}_c^{add} + \mathbf{1}_c$, we obtain the (unlumped) multiplicative coarse-level problem

$$\mathbf{RAPx}_c = \mathbf{0}_c.$$

Moreover, by making the same substitution into the additive coarse-grid correction (3.29) we observe that

$$\mathbf{x}^{\text{CGC}} = \bar{\mathbf{x}}^{(k)} + \mathbf{Pe}_c^{add} = \mathbf{P}(\mathbf{1}_c + \mathbf{e}_c^{add}) = \mathbf{Px}_c,$$

which is equivalent to the multiplicative coarse-grid correction (3.5). □

The analysis in Proposition 3.6.1 can be extended to the multilevel case provided the relaxed solution $\bar{\mathbf{x}}^{(k)}$ lies exactly in the range of interpolation on all levels. Whereas the multiplicative scheme can converge only when $\mathbf{x}$ lies exactly in the range of interpolation, the additive scheme can converge if $\mathbf{x}$ and $\mathbf{e}^{add}$ are only approximately in the range of interpolation. This observation motivates the following approach. We first adaptively determine transfer operators on all levels by performing a few multiplicative cycles. We then freeze the transfer and (lumped) coarse-level operators, and use additive cycles to update the solution. Essentially, the multiplicative cycles form a setup phase and the additive cycles form a solve phase. Additive cycles are computationally much cheaper than multiplicative cycles because the transfer and coarse-level operators are not computed on each level.

Therefore, this hybrid method is potentially much cheaper than the standalone multiplicative method. We note that since the two-level MCAMG method modifies the coarse-level system operator through a lumping process (see §3.4), the equivalence in Proposition 3.6.1 does not hold. In particular,

$$(\mathbf{RAP} + \tilde{\mathbf{S}}_c)\mathbf{x}_c = \mathbf{0}_c \quad \Leftrightarrow \quad \mathbf{RAP}\mathbf{x}_c = -\tilde{\mathbf{S}}_c\mathbf{x}_c \tag{3.30}$$

and the coarse-level problems are not equivalent. As the fine-level solution converges, by the equation for the multiplicative coarse-grid correction (3.5) we have that $\mathbf{x}_c \to \mathbf{1}_c$, hence the right-hand side $\tilde{\mathbf{S}}_c\mathbf{x}_c \to \mathbf{0}_c$. Therefore, the equivalence between the lumped multiplicative two-level scheme and the additive two-level scheme should improve as the multiplicative method converges. Regardless, the hybrid method should still work well if the smooth additive error components are approximated well by the range of interpolation. Naturally, if the convergence rate of the additive cycles begins to deteriorate, signifying that the smooth additive error components are not approximated well enough by the range of interpolation, a multiplicative cycle can be used to update the transfer operators and coarse-level operators as well as the solution.

Additive cycles for solving $\mathbf{Ax} = \mathbf{0}$ are given by Algorithm 2.2 with transfer operators and coarse-level system operators on each level constructed in the multiplicative cycles. Since the coarsest-level problem $\mathbf{A}_L\mathbf{x}_L = \mathbf{f}_L$ is singular, care must be taken when computing its solution. As discussed in [118], the solution on the coarsest level should contain as little of the nullspace of $\mathbf{A}_L$ as possible to avoid contaminating the coarse-grid correction with unwanted nullspace components on finer levels. Consider a two-level method. If the coarse-level solution contains some proportion of the nullspace of $\mathbf{A}_c$ then the correction $\mathbf{P}\mathbf{x}_c$ may contain some proportion of the nullspace of $\mathbf{A}$, that is, the sought after vector $\mathbf{x}$. In this case the coarse-grid correction $\bar{\mathbf{x}}^{(k)} + \mathbf{P}\mathbf{x}_c$ may actually subtract away some proportion of $\mathbf{x}$ from $\bar{\mathbf{x}}^{(k)}$, and consequently ruin the computed solution. To obtain a unique "null-free" solution on the coarsest level the authors of [118] advocate setting $\mathbf{x}_L = \mathbf{A}_L^{\dagger}\mathbf{f}_L$, where the Moore–Penrose pseudoinverse of $\mathbf{A}_L$ is computed via the singular value decomposition of $\mathbf{A}_L$. Since $\text{null}(\mathbf{A}_L) = \text{span}(\mathbf{v}_{n_L})$, where $\mathbf{v}_{n_L}$ is the right singular vector of $\mathbf{A}_L$ corresponding to its zero singular value, it follows that

$$\text{range}(\mathbf{A}_L^\dagger) \cap \text{null}(\mathbf{A}_L) = \emptyset.$$

Additive and multiplicative cycles are combined into a simple hybrid solver referred to as MCAMG-hybrid (Algorithm 3.2). The first if-then-else block determines if the multigrid hierarchy constructed by the multiplicative cycles is sufficiently accurate to warrant using additive cycles. In practice we use a setup tolerance of $\tau = 10^{-4}$. There are two conditional statements in the second if-then-else block. The first conditional statement assesses whether the additive cycles have stagnated, and consequently if a multiplicative cycle should be used to update the transfer and coarse-level system operators.

Algorithm 3.2: Hybrid multiplicative-additive solver for Markov chains

```
   **Input**: Initial guess $\mathbf{x}^{(0)}$, $\mathbf{A}$, setup tolerance $\tau \in (0,1)$, positivity tolerance
              $\delta \in (0,1)$, stagnation factor $C \in (0,1]$
   **Output**: Converged iterate $\mathbf{x}^{(k+1)}$
1. Set $k \leftarrow 0$
2. if $\|\mathbf{A}\mathbf{x}^{(k)}\|_1 > \tau$ then
3.     Obtain $\mathbf{x}^{(k+1)}$ via one MCAMG cycle with initial guess $\mathbf{x}^{(k)}$
   else
4.     Obtain $\mathbf{x}^{add}$ via one additive cycle with initial guess $\mathbf{x}^{(k)}$
       if $\|\mathbf{A}\mathbf{x}^{add}\|_1 \geq C\|\mathbf{A}\mathbf{x}^{(k)}\|_1$ or $\max_i - x_i^{add} > \delta$ then
5.         Obtain $\mathbf{x}^{(k+1)}$ via one MCAMG cycle with initial guess $\mathbf{x}^{(k)}$
       else
6.         Set $x_i^{(k+1)} \leftarrow |x_i^{add}|$ for all $i$ and normalize
       end
   end
7. If the stopping criterion is satisfied return $\mathbf{x}^{(k+1)}$, otherwise set $k \leftarrow k+1$ and go to 2
```

Varying C over the interval $(0, 1]$ controls how aggressive the check is for stagnation. The numerical tests in §3.7 use the least aggressive setting $C = 1$, that is, $\mathbf{x}^{add}$ is accepted as the new iterate as long as its residual one-norm is less than that of the previous iterate.

The second conditional statement checks the magnitude of negative values in the additive solution. Since we cannot guarantee positivity of the additive cycle iterates, small negative values may be present in $\mathbf{x}^{add}$. If the negative components in a vector $\mathbf{u} \in \mathbb{R}^n$ are sufficiently small in magnitude, then the residual one-norm is essentially unaffected by taking the componentwise absolute value, as shown by the following inequality:

$$\begin{aligned}
\big| \|\mathbf{Au}\|_1 - \|\mathbf{A}|\mathbf{u}|\|_1 \big| \le \|\mathbf{Au} - \mathbf{A}|\mathbf{u}|\|_1 &= \sum_i \Big| \sum_j a_{ij}(u_j - |u_j|) \Big| \\
&= 2 \sum_i \Big| \sum_{j \in \mathcal{J}^-} a_{ij} u_j \Big|, \quad \mathcal{J}^- = \{ j \,:\, u_j < 0 \} \\
&\le 2 \sum_i \sum_{j \in \mathcal{J}^-} |a_{ij}||u_j| \\
&\le 2\delta \sum_i \sum_{j \in \mathcal{J}^-} |a_{ij}|, \quad \max_j -u_j < \delta \\
&\le 2\delta \sum_{j \in \mathcal{J}^-} \sum_i |a_{ij}| \\
&\le 4\delta \sum_{j \in \mathcal{J}^-} a_{jj}. \qquad (3.31)
\end{aligned}$$

Accordingly, if all negative values in $\mathbf{x}^{add}$ are sufficiently small in magnitude (smaller than some tolerance $0 < \delta \ll 1$) we accept $\mathbf{x}^{add}$ and take its absolute value. In practice we use the tolerance $\delta = 10^{-20}$. The hybrid algorithm presented here is a simple version of the more sophisticated on-the-fly (OTF) adaptive AMG algorithm for Markov chains proposed in [118]. Essentially, the OTF framework tries to maximize its use of additive cycles during the setup phase as well as during the solve phase to obtain fast convergence speeds. Further sophistication is obtained by overlapping the setup and solution phases in a parallel setting. The OTF framework [118] has also been considered for Markov chain problems in [53]. We note that the combination of an adaptive multiplicative setup phase followed by additive cycles is not new and has been considered extensively in the literature [20, 21, 26, 29, 30, 31].

3.7 Numerical results

In this section we present the results of numerical tests. All experiments are performed using Matlab version 7.5.0.342 (R2007b), where every attempt has been made to obtain optimized performance by exploiting sparse data types and vectorization in Matlab, and by implementing MEX (Matlab Executables) in the C programming language for the bottleneck operations in the multilevel methods. Timings are reported for a laptop running Windows XP, with a 2.50 GHz Intel Core 2 Duo processor and 4 GB of RAM. For the stopping criterion we iterate until the one-norm of the residual has been reduced by a factor of 10^{12}, that is, until

$$\frac{\|\mathbf{A}\mathbf{x}^{(k)}\|_1}{\|\mathbf{x}^{(k)}\|_1} < 10^{-12}\|\mathbf{A}\mathbf{x}^{(0)}\|_1. \tag{3.32}$$

The initial guess $\mathbf{x}^{(0)}$ is randomly generated by sampling the standard uniform distribution and then normalizing with respect to the one-norm.

The numerical tests compare the standalone MCAMG method (Algorithm 3.1) with the hybrid multiplicative-additive method (Algorithm 3.2) denoted by MCAMG-hybrid, or MCAMG-h in the tables. We also consider MCAMG-frozen in which $\tilde{\mathbf{P}}$ has been frozen on all levels after two iterations. The parameters for these methods are given in Table 3.1. For MCAMG and MCAMG-frozen we use $\mathrm{V}(2,2)$-cycles (two pre- and post-relaxations

Parameter	Value
Strength of connection parameter θ	0.25
Lumping parameter η	0.01
Weighted Jacobi relaxation parameter ω	0.7
Maximum number of points on coarsest level	20
Maximum number of levels	20

Table 3.1: Parameters for MCAMG and MCAMG-hybrid methods.

per level), and for the MCAMG-hybrid method we use $\mathrm{V}(4,2)$ setup cycles and $\mathrm{V}(1,1)$ solution cycles. We note that any additional MCAMG cycles during the solution phase of MCAMG-hybrid (as in line 3 of Algorithm 3.1) are V$(2,2)$-cycles. We note that in the hybrid method a larger number of relaxations is required by the setup cycles to ensure that sufficiently accurate transfer operators and coarse-level system operators are obtained prior to the additive solve phase. Moreover, because the computational cost of a setup cycle is significantly more than the cost of a solution cycle, it is worth performing the extra relaxations (which are cheap relative to the entire setup cycle cost) in order to minimize the number of setup cycles.

In the tables below we report the problem size on the finest level (n), the number of nonzero elements in the finest-level operator (nnz), and the following performance measures.

1. The number of iterations (it).
2. The total execution time in seconds (time).
3. The operator complexity (C_{op}) on the last cycle; see §2.6.3.
4. The grid complexity (C_{grid}) on the last cycle; see §2.6.3.
5. The number of levels on the last cycle (levs).
6. The convergence factor $\gamma := \gamma^{(k)}$ with $k_0 = 5$; see (2.62).
7. The lumping ratio (R_{lump}) on the last cycle, which is defined as

$$R_{lump} = \frac{\sum_{\ell=0}^{L-1} (\text{number of offending index pairs in the unlumped matrix } \mathbf{A}_{\ell+1})}{\sum_{\ell=0}^{L} \text{nnz}(\mathbf{A}_\ell)}.$$

We note that while the operator complexity, grid complexity, number of levels, and lumping ratio may vary during the initial iterations, as the solution converges the multilevel hierarchy should stabilize and these quantities should approach constant values. In the case of the hybrid method these quantities are determined by the last multiplicative cycle performed either during the setup or solve phase. The lumping ratio gives the fraction of matrix elements for which lumping is required, and is thus an indication of the extra work required for lumping. We note that lumping is not required in the finest-level matrix, so lumping only contributes extra work starting from the second level.

The first test problems we consider correspond to discrete-time Markov chains that arise from a *random walk* on a weighted undirected graph. We assume the underlying graph is connected and the edge weights w_{ij} are positive. The transition probability matrix $\mathbf{B}$ for these Markov chains is defined as follows. Let $\mathbf{C}$ be the *adjacency matrix* of the weighted graph, that is, $c_{ij} = w_{ij}$ for each edge $\{i, j\}$, and note that $\mathbf{C}$ is symmetric. Let $\mathbf{D} = \operatorname{diag}(\mathbf{d})$ where $d_i = \sum_k w_{ik}$ is the sum of the weights of all outgoing edges from node i. Then $\mathbf{B} = \mathbf{C}\mathbf{D}^{-1}$. The transition probability matrix $\mathbf{B}$ is similar to a symmetric adjacency matrix:

$$\mathbf{D}^{-1/2}\mathbf{B}\mathbf{D}^{1/2} = \mathbf{D}^{-1/2}\mathbf{C}\mathbf{D}^{-1/2},$$

hence $\mathbf{B}$ has a real spectrum. Moreover, $\mathbf{x} = \mathbf{d}/(\mathbf{1}^\top\mathbf{d})$ is the stationary distribution of $\mathbf{B}$ because

$$(\mathbf{B}\mathbf{x})_i = \sum_j b_{ij}x_j = \sum_j \frac{w_{ij}}{d_j}\frac{d_j}{\mathbf{1}^\top\mathbf{d}} = \frac{d_i}{\mathbf{1}^\top\mathbf{d}} = x_i \quad \text{for all} \quad i. \tag{3.33}$$

Note that

$$\bar{\mathbf{A}} = \mathbf{A}\operatorname{diag}(\mathbf{x}) = \frac{1}{\mathbf{1}^\top\mathbf{d}}(\mathbf{I} - \mathbf{B})\operatorname{diag}(\mathbf{d}) = \frac{1}{\mathbf{1}^\top\mathbf{d}}(\mathbf{D} - \mathbf{C}), \tag{3.34}$$

which implies that $\bar{\mathbf{A}}$ is symmetric when $\mathbf{x}$ is the stationary distribution. While these test problems are academic in nature, they are nonetheless instructive because of their similarity to linear systems arising from the discretization of partial differential equations, which are well understood in the context of AMG. Furthermore, these test problems give an indication of the kind of performance that can be expected for more general problems.

3.7.1 Isotropic two-dimensional lattice

The first test problem we consider is an isotropic two-dimensional (2D) lattice (Figure 3.1). Results for the isotropic 2D lattice are given in Table 3.2. As expected we observe near-

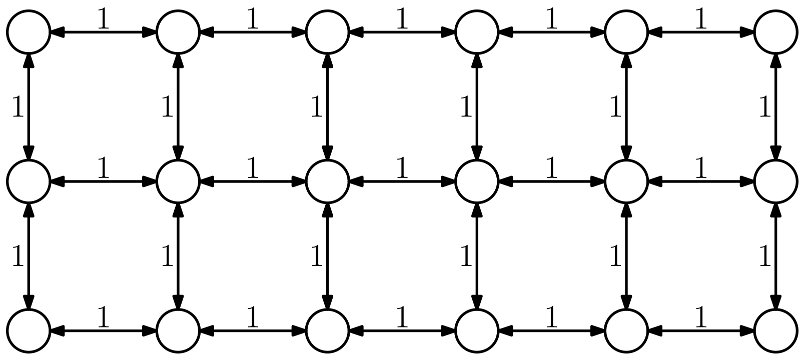


Figure 3.1: Graph of isotropic 2D lattice. All weights are equal to one.

optimal performance, that is, small convergence factors and bounded operator complexities that do not grow as n increases. Figure 3.2 shows the execution time scaling for MCAMG and MCAMG-frozen. We observe that by freezing the transfer operators and coarse-level operators after a few cycles, the MCAMG execution times can effectively be cut in half while maintaining scalability. We note that there was no lumping on the last cycle, which is common for problems in which $\mathbf{A}$ is similar to a symmetric matrix.

Method	n	nnz	it	γ	time	C_{op}	C_{grid}	R_{lump}	levs
MCAMG	4096	20224	11	0.10	0.4	2.20	1.68	0	6
	16384	81408	11	0.10	1.5	2.20	1.67	0	7
	65536	326656	11	0.10	6.6	2.20	1.67	0	8
	262144	1308672	11	0.10	28.5	2.20	1.67	0	9
	589824	2946048	11	0.10	65.0	2.20	1.67	0	10
MCAMG-frozen	4096	20224	11	0.10	0.2	2.20	1.68	0	6
	16384	81408	11	0.10	0.9	2.20	1.67	0	7
	65536	326656	11	0.10	3.6	2.20	1.67	0	8
	262144	1308672	11	0.10	15.6	2.20	1.67	0	9
	589824	2946048	11	0.10	35.8	2.20	1.67	0	10

Table 3.2: Isotropic 2D lattice. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}.

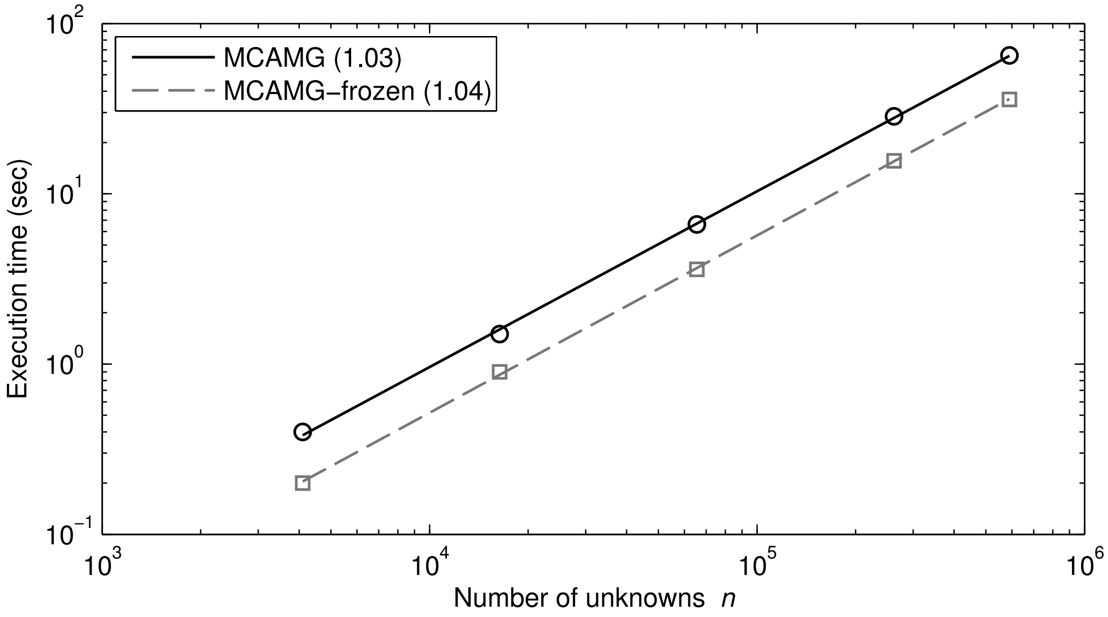


Figure 3.2: Isotropic 2D lattice execution time scaling of MCAMG and MCAMG-frozen. The solid and dashed lines are the best-fit lines through the data points (circles and squares). Numerical values in the legend are the slopes of the best-fit lines.

3.7.2 Anisotropic two-dimensional lattice

The next test problem is an anisotropic 2D lattice with small edge weights $\varepsilon = 10^{-6}$ in the y-direction (Figure 3.3). Anisotropic grids are difficult for standard geometric multigrid

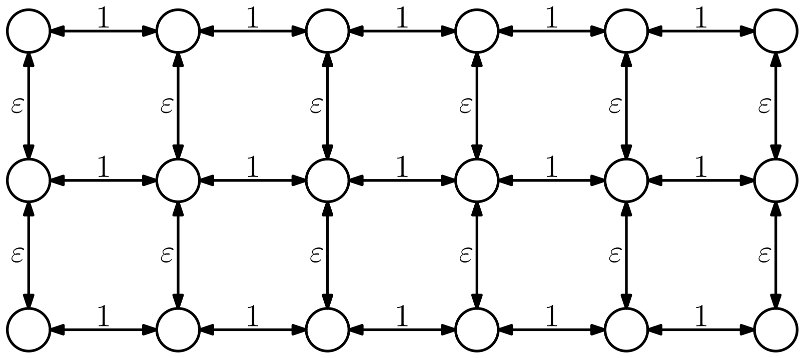


Figure 3.3: Graph of anisotropic 2D lattice with small edge weights $\varepsilon = 10^{-6}$.

because the (pointwise) relaxation scheme is unable to smooth in the direction of weak connections. The remedy is to employ either semicoarsening with pointwise relaxation or line relaxation with full coarsening [32, 119]. Classical AMG applied to anisotropic grid problems naturally produces semicoarsened grids by coarsening only in the direction of strong dependence [32]. In addition, smoothed aggregation multigrid with strength-based aggregation is also an effective method for anisotropic problems [49]. Results for the anisotropic 2D lattice are given in Table 3.3. Similar to the isotropic test case we observe near-optimal performance for the anisotropic 2D lattice (Figure 3.4). Approximately one and half times as many levels are required for the anisotropic test case as for the isotropic test case because of the coarsening routine's automatic semicoarsening in the direction of strong connections. Again we note that there was no lumping on the last cycle, which is common for problems in which **A** is similar to a symmetric matrix.

Method	n	nnz	it	γ	time	C_{op}	C_{grid}	R_{lump}	levs
	4096	20224	10	0.08	0.3	2.67	2.00	0	9
	16384	81408	10	0.08	1.1	2.73	2.00	0	11
MCAMG	65536	326656	10	0.08	4.7	2.76	2.00	0	13
	262144	1308672	10	0.08	20.3	2.78	2.00	0	15
	589824	2946048	10	0.08	46.9	2.78	2.00	0	16
	4096	20224	10	0.08	0.2	2.67	2.00	0	9
	16384	81408	10	0.08	0.7	2.73	2.00	0	11
MCAMG-frozen	65536	326656	11	0.08	3.5	2.76	2.00	0	13
	262144	1308672	11	0.08	15.3	2.78	2.00	0	15
	589824	2946048	11	0.08	35.7	2.78	2.00	0	16

Table 3.3: Anisotropic 2D lattice. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}.

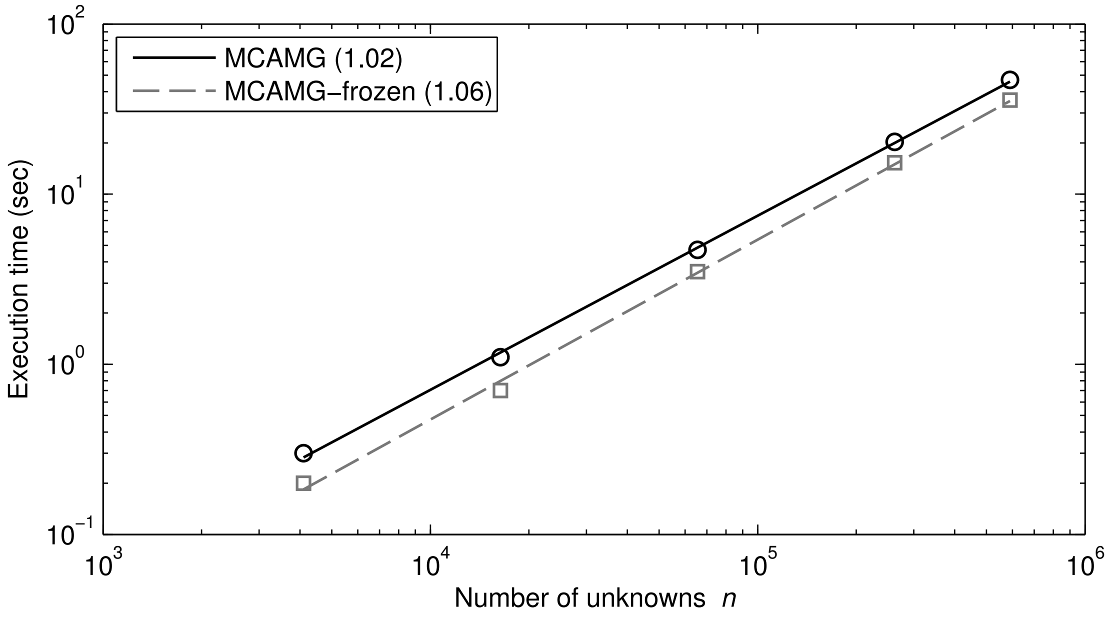


Figure 3.4: Anisotropic 2D lattice execution time scaling of MCAMG and MCAMG-frozen. The solid and dashed lines are the best-fit lines through the data points (circles and squares). Numerical values in the legend are the slopes of the best-fit lines.

The remaining test problems we consider include a tandem queueing network [110], a random walk on an unstructured directed planar graph [52], an octagonal mesh problem [50], a stochastic Petri net problem [69, 92], an ATM queueing network model [100], and a reliability model [8, 110]. We note that the fifth test case arises from Markov chain applications, and is from the MARCA (Markov chain analyzer) collection [109]. These six test problems are intended to be challenging in that they are nonsymmetric with complex spectra, and have eigenvalues that tend to cluster near $\lambda = 1$. Moreover, the MARCA test problem is an example of a nearly completely decomposable Markov chain [91]. The transition matrix spectra of the test problems are plotted in Figure 3.5. We note that the spectrum in subplot (f) contains only real eigenvalues.

To further characterize the spectra of these test problems we investigate the asymptotic relationship between the subdominant eigenvalue of the transition matrix $\mathbf{B}$ and the

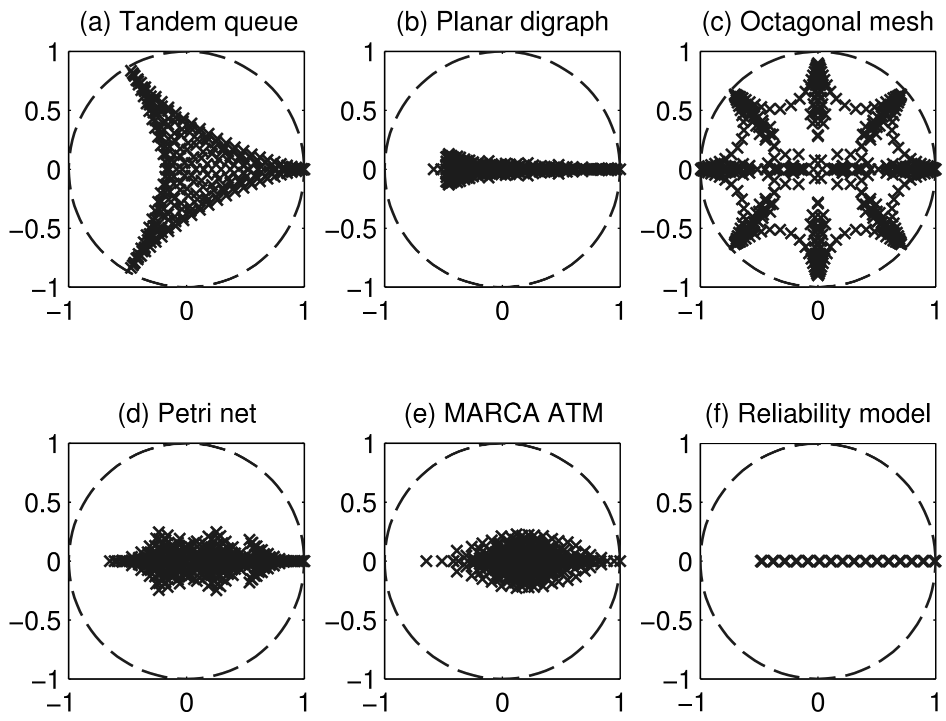


Figure 3.5: Transition matrix spectra of MCAMG test problems.

number of states n. In particular, we look for the following relationship:

$$1 - \mathrm{Re}(\lambda_2) \approx \beta \left(\frac{1}{n}\right)^{\alpha} \quad \text{as} \quad n \to \infty, \tag{3.35}$$

where α and β are positive constants, and λ_2 is the subdominant eigenvalue of $\mathbf{B}$ with real part closest to one. The exponent α determines how rapidly the subdominant eigenvalue approaches unity as n grows large. Log-log plots in Figure 3.6 illustrate the asymptotic behavior described in (3.35). An estimate of α may also provide insight into the rate at which traditional stationary iterative methods converge. In particular, we expect multilevel methods to outperform traditional stationary methods as the subdominant eigenvalue of $\mathbf{B}$ with largest real part approaches unity. Although we cannot prove this fact (nor do we know of any proof), in general the weighted Jacobi preconditioner $\mathbf{M}^{-1} = \omega\mathbf{D}^{-1}$ appears unable to significantly counteract poor scaling in the transition matrix $\mathbf{B}$. Log-log plots in Figure 3.7 illustrate the asymptotic behavior between the subdominant eigenvalue with real part closest to one and the number of states n for the weighted Jacobi iteration matrix $\mathbf{I} - \omega\mathbf{D}^{-1}\mathbf{A}$ with $\omega = 0.7$ and $\mathbf{A} = \mathbf{I} - \mathbf{B}$. Figure 3.8 illustrates the nonzero structure of the test cases. We note that the octagonal mesh and reliability model have symmetric nonzero structure, whereas the other test cases are fully nonsymmetric.

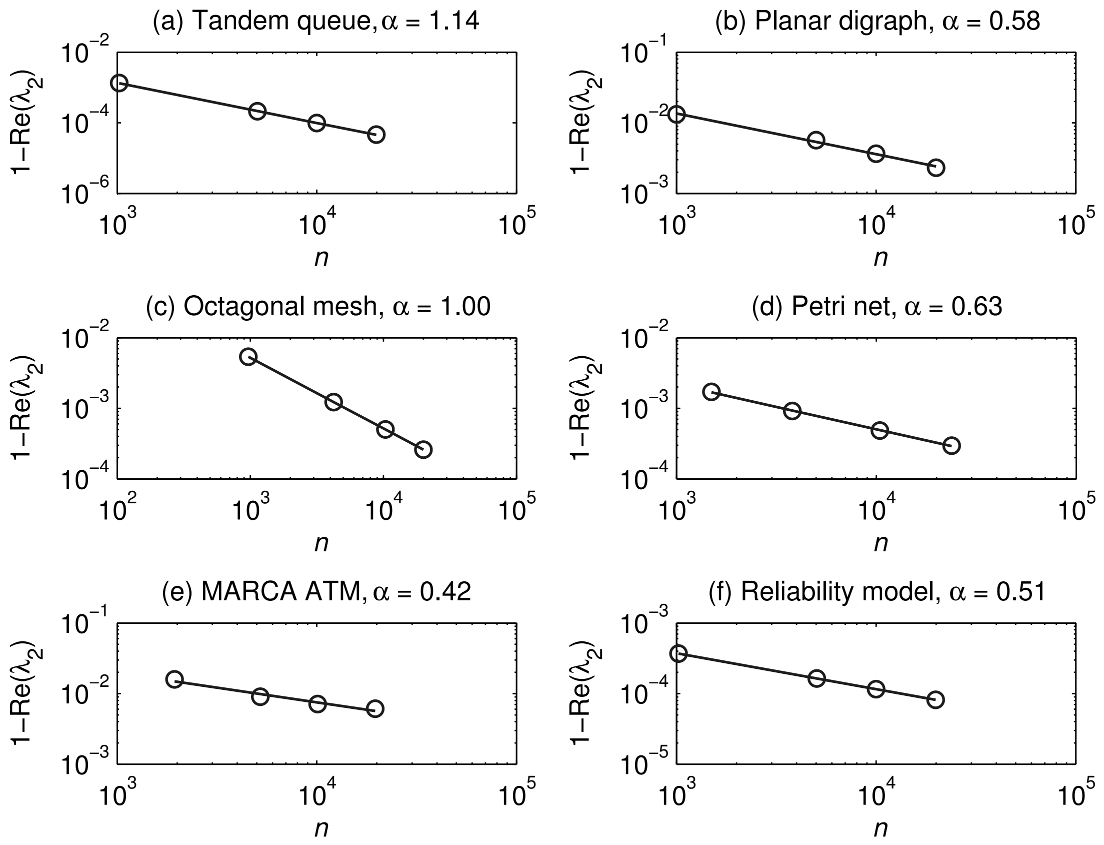


Figure 3.6: Asymptotic behaviour of the eigenvalue with maximum real part associated with the transition matrix $\mathbf{B}$.

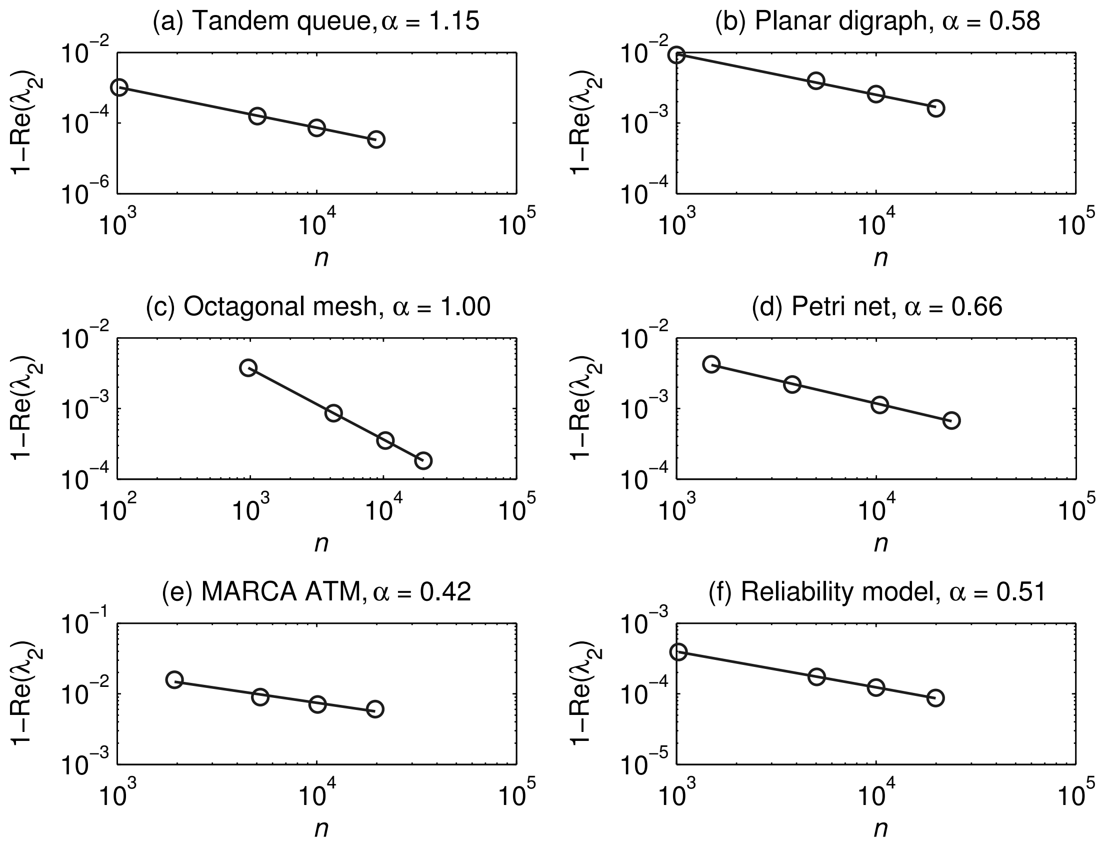


Figure 3.7: Asymptotic behavior of the subdominant eigenvalue with maximum real part corresponding to the weighted Jacobi iteration matrix $\mathbf{I} - \omega \mathbf{D}^{-1}\mathbf{A}$ with $\omega = 0.7$.

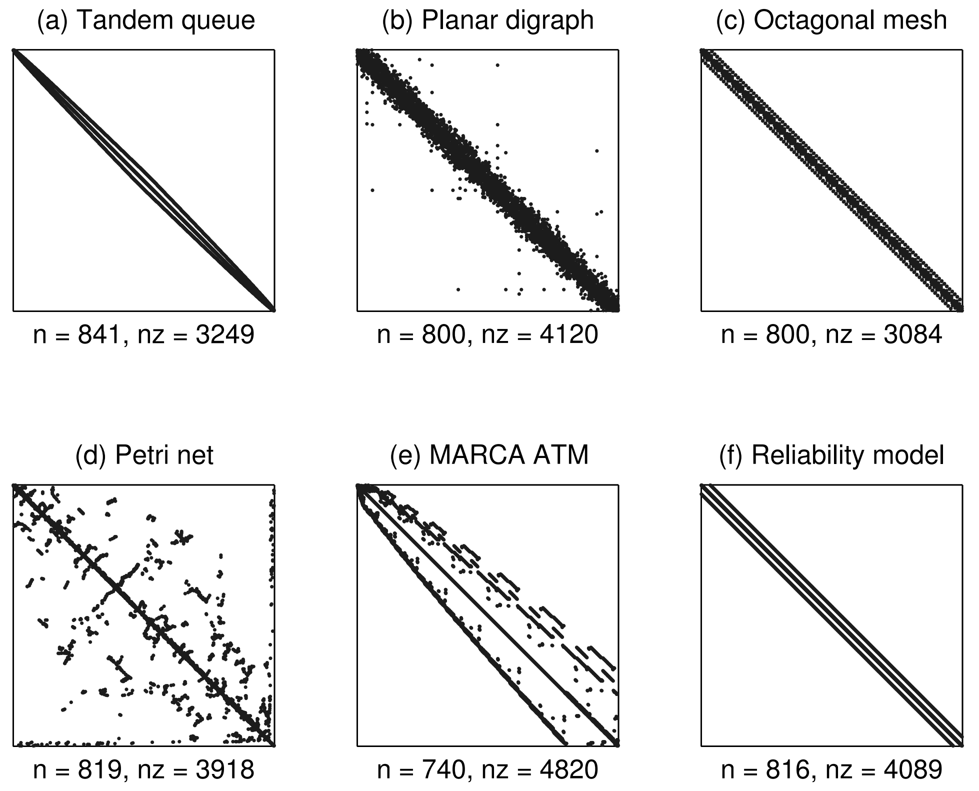


Figure 3.8: Nonzero structure of $\mathbf{A} = \mathbf{I} - \mathbf{B}$ for the MCAMG test problems.

3.7.3 Tandem queueing network

The first test problem we consider is the tandem queueing network from [110], in which 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. The states of the system are represented by tuples (n_1, n_2), where n_i is the number of customers waiting in the ith queue. Thus, if the capacity of each queue is N customers, the model has $(N+1)^2$ states in total. We choose $(\mu,\ \mu_1,\ \mu_2) = (10, 11, 10)$ for the arrival and service time rates, which leads to slow mixing. The possible transitions and the rates at which they occur are given by

$$\begin{aligned} (n_1, n_2) &\to (n_1 + 1, n_2) && \text{with rate} \quad \mu, \\ (n_1, n_2) &\to (n_1 - 1, n_2 + 1) && \text{with rate} \quad \mu_1, \\ (n_1, n_2) &\to (n_1, n_2 - 1) && \text{with rate} \quad \mu_2, \end{aligned}$$

for $0 \le n_1, n_2 \le N$. If we order the states such that state (i, j) has index $(N+1)(N-i) + N - j + 1$, then the resulting $(N+1)^2 \times (N+1)^2$ column-oriented infinitesimal generator matrix $\mathbf{Q}$ is block tridiagonal with stencil $\mathbf{Q} = [\mathbf{B},\ \mathbf{A},\ \mathbf{C}]$, where

$$\mathbf{A} = \begin{bmatrix} * & & & & \\ \mu_2 & * & & & \\ & \mu_2 & * & & \\ & & \ddots & \ddots & \\ & & & \mu_2 & * \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & \mu_1 & & & \\ & 0 & \mu_1 & & \\ & & 0 & \ddots & \\ & & & \ddots & \mu_1 \\ & & & & 0 \end{bmatrix}, \quad \text{and} \quad \mathbf{C} = \mu\mathbf{I}.$$

The diagonal elements indicated by asterisks are the negated sums of the off-diagonal elements in their corresponding columns in $\mathbf{Q}$.

The results for the tandem queueing network are given in Table 3.4. While the operator complexities appear to be bounded independent of the problem size, they are somewhat larger than we would like. Unfortunately, attempts to reduce the operator complexity through truncation of the interpolation operator [119] or increasing the strength threshold were unsuccessful. Preliminary experiments with aggressive coarsening [119] were also conducted, however, the computation of the first coarse level was too expensive for the larger problems ($n \geq 262144$) to be of any use. Although the hybrid method effectively reduces the execution time by half for $n = 589824$, its convergence rates are somewhat higher than those of MCAMG, and it scales slightly worse than MCAMG in terms of execution time. The performance of MCAMG-hybrid compared with MCAMG is not surprising given the approximate equivalence of the additive and multiplicative cycles (as discussed in §3.6), and to a lesser degree because the hybrid method solution cycles are V$(1, 1)$-cycles whereas the MCAMG cycles are V$(2, 2)$-cycles. In general, we expect the convergence rate of the hybrid method at the very best to equal that of MCAMG.

Method	n	nnz	it	γ	time	C_{op}	C_{grid}	R_{lump}	levs
MCAMG	4096	16129	16	0.21	0.8	4.47	2.13	0.085	7
	16384	65025	18	0.27	3.6	4.54	2.13	0.086	9
	65536	261121	24	0.42	20.3	4.61	2.12	0.080	11
	262144	1046529	25	0.43	91.3	4.65	2.12	0.067	13
	589824	2356225	21	0.35	180.3	4.67	2.13	0.067	14
MCAMG-h	4096	16129	19(0)	0.32	(0.2, 0.1)	4.47	2.13	0.123	7
	16384	65025	22(0)	0.38	(0.7, 0.3)	4.53	2.12	0.138	9
	65536	261121	33(1)	0.53	(2.8, 3.1)	4.60	2.12	0.134	11
	262144	1046529	30(3)	0.50	(15.5, 20.0)	4.64	2.12	0.062	13
	589824	2356225	26(3)	0.46	(35.6, 43.5)	4.65	2.12	0.073	14

Table 3.4: Tandem queueing network. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. The bracketed times (t_0, t_1) are the setup and solve times in seconds. A bracketed value beside an iteration count is the number of additional multiplicative cycles performed during the solve phase, which is included in the overall iteration count.

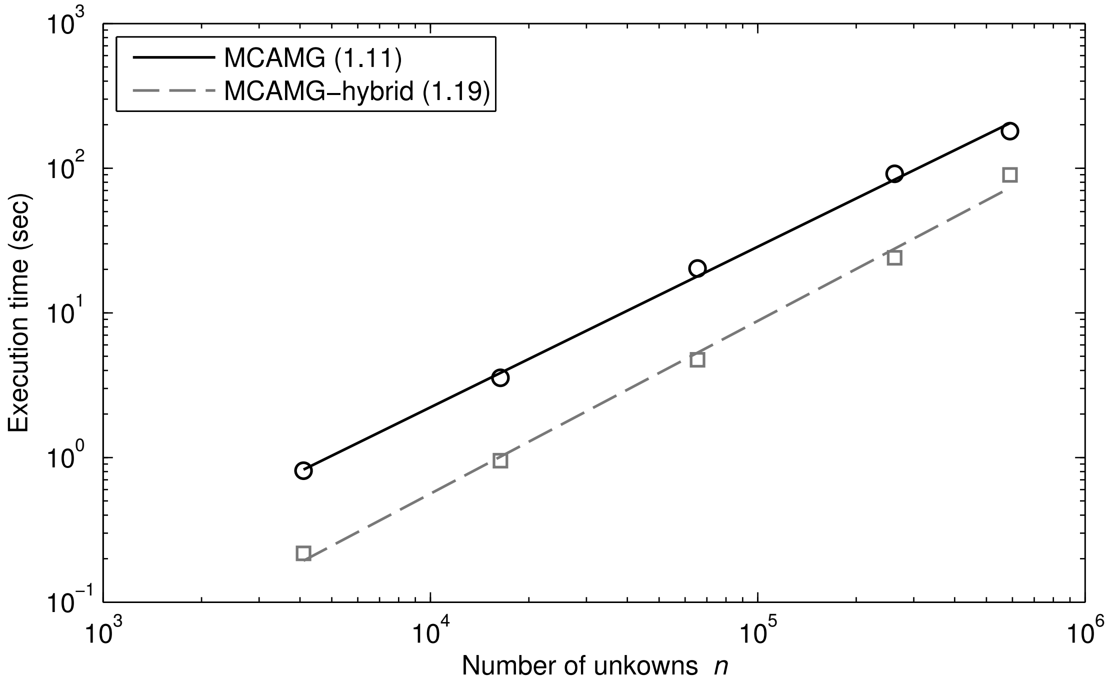
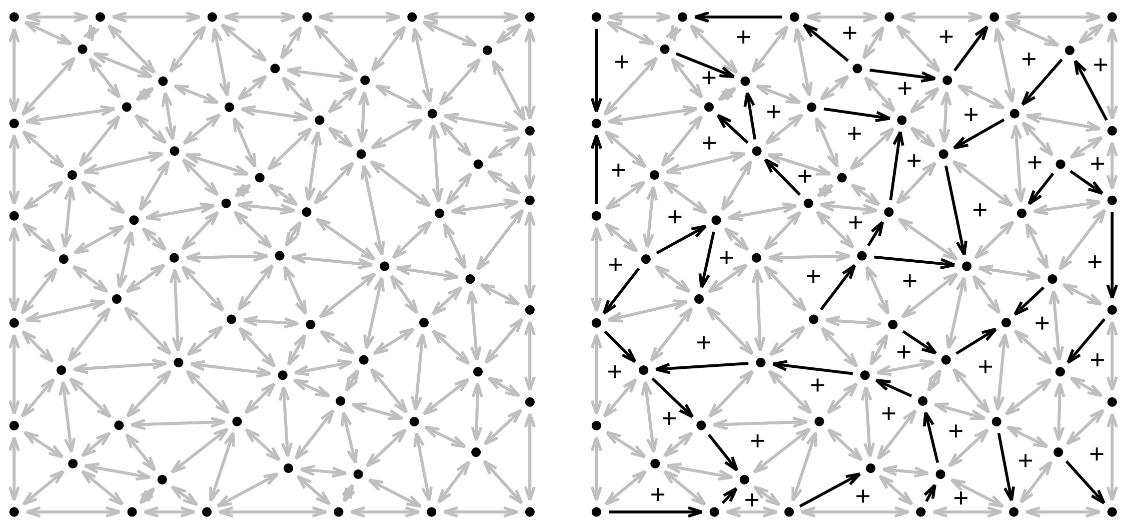


Figure 3.9: Tandem queueing network execution time scaling of MCAMG and MCAMG-hybrid. The solid and dashed lines are the best-fit lines through the data points (circles and squares). Numerical values in the legend are the slopes of the best-fit lines.

3.7.4 Unstructured planar graph

The next test problem we consider is a random walk on an unstructured directed planar graph [52]. To construct the graph we randomly distribute n points in the unit square $[0, 1] \times [0, 1]$. These points are then connected via Delaunay triangulation, which yields an undirected planar graph $\mathcal{G}$. To obtain a directed graph $\mathcal{D}$ from $\mathcal{G}$ we proceed as follows. A subset of triangles is selected from the triangulation such that no two triangles in the set share an edge. This subset is constructed by selecting an unmarked triangle, marking it with a "+", and then marking its neighbors with a "−". This process is repeated for the next unmarked triangle until all triangles are marked. Then, one edge on each "+" triangle is randomly made unidirectional. We note that while some of the "−" triangles will also have unidirectional edges, each "+" triangle will have one and only one unidirectional edge.

This process ensures that $\mathcal{D}$ is strongly connected, or equivalently, that the corresponding Markov chain is irreducible. Figure 3.10 illustrates a typical planar graph arising from this construction.



(a) Original graph $\mathcal{G}$ (b) Directed graph $\mathcal{D}$

Figure 3.10: Unstructured directed planar graph. The black dots represent nodes, and the light gray arrows represent edges in the original planar graph $\mathcal{G}$. The black arrows represent unidirectional edges, and triangles marked by "+" have a single edge that was made unidirectional.

Table 3.5 gives the results for the unstructured directed planar graph problem. Again we observe near-optimal performance of the MCAMG method. In this case the hybrid method results in much faster execution times with a speedup of four times for $n = 262144$, and scaling that is almost as good as MCAMG. The execution time scaling of the two methods is given in Figure 3.11. This test case is a prime example of the classical AMG coarsening routine's ability to robustly handle highly unstructured problems in which geometric coarsening is impractical.

Method	n	nnz	it	γ	time	C_{op}	C_{grid}	R_{lump}	levs
MCAMG	4096	25402	21	0.37	0.9	2.73	1.69	0.003	7
	16384	101737	23	0.40	4.0	2.80	1.69	0.003	9
	65536	407175	23	0.40	17.5	2.82	1.69	0.003	10
	262144	1628955	25	0.45	81.9	2.82	1.69	0.003	11
MCAMG-h	4096	25402	27(0)	0.47	$(0.1, 0.1)$	2.72	1.69	0.003	7
	16384	101737	27(0)	0.47	$(0.6, 0.4)$	2.80	1.69	0.003	9
	65536	407175	39(0)	0.61	$(2.3, 2.6)$	2.82	1.68	0.003	10
	262144	1628955	33(0)	0.56	$(10.1, 9.6)$	2.83	1.69	0.003	12

Table 3.5: Unstructured directed planar graph. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. The bracketed times (t_0, t_1) are the setup and solve times in seconds. A bracketed value beside an iteration count is the number of additional multiplicative cycles performed during the solve phase.

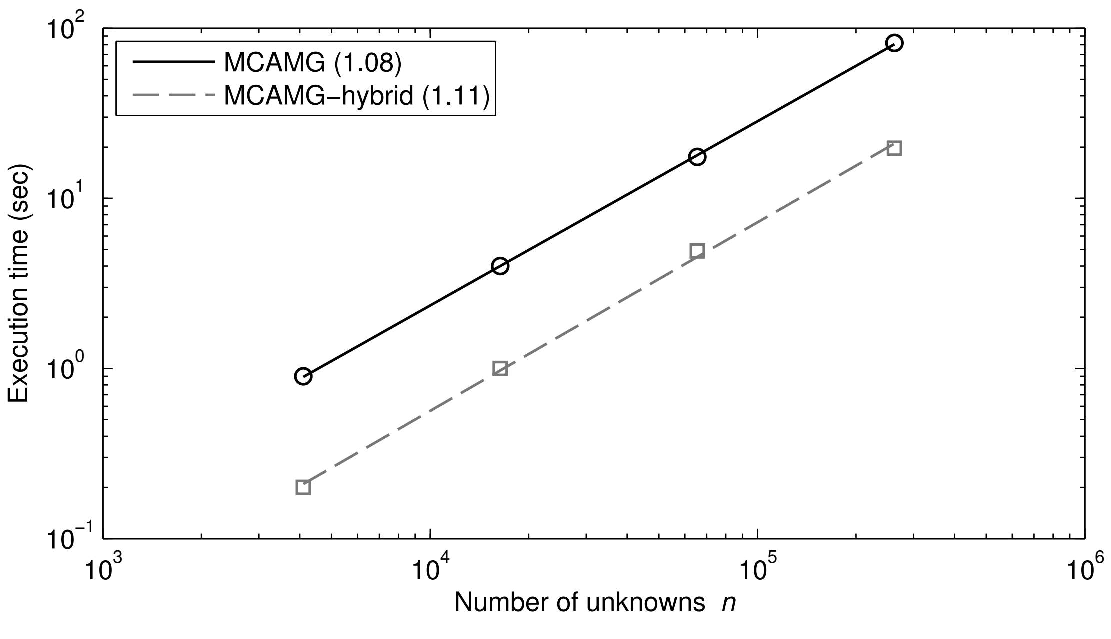


Figure 3.11: Unstructured directed planar graph execution time scaling of MCAMG and MCAMG-hybrid. The solid and dashed lines are the best-fit lines through the data points (circles and squares). Numerical values in the legend are the slopes of the best-fit lines.

3.7.5 Octagonal mesh

The next test problem we consider features elements of web traffic modeling restricted to the directed planar graph $\mathcal{G}$ illustrated in Figure 3.12. This test problem is designed so the spectrum of the transition matrix uniformly fills the unit circle in the complex plane (see subplot (c) in Figure 3.5). The directed graph $\mathcal{G}$ has $N = 8n_xn_y$ nodes, where n_x is the

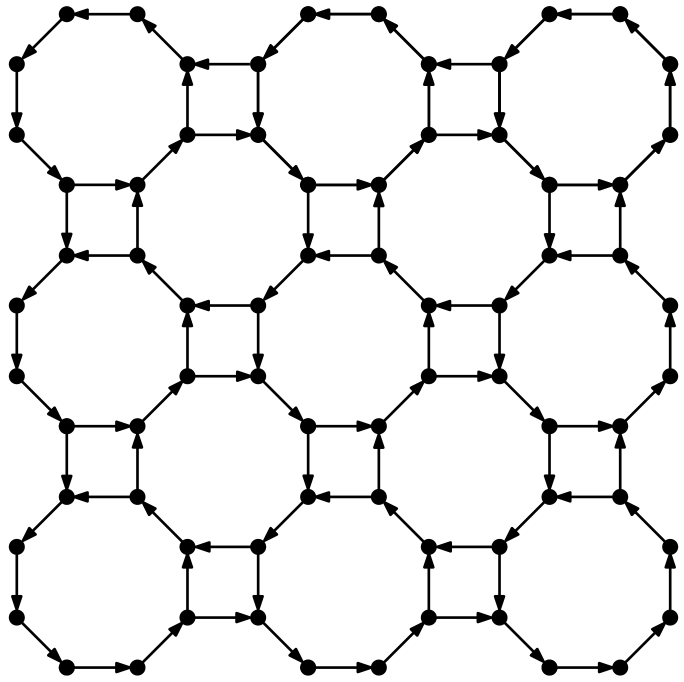


Figure 3.12: Graph $\mathcal{G}$ of the octagonal mesh with $n_x = n_y = 3$.

number of octagons in the x-direction and n_y is the number of octagons in the y-direction. To each node in $\mathcal{G}$ we assign the following probabilities:

μ_+	The probability of moving forward, distributed evenly among outgoing arcs.
μ_0	The probability of staying at the current node.
μ_-	The probability of moving backward, distributed evenly among incoming arcs.

These probabilities are defined so that $\mu_0 + \mu_+ + \mu_- = 1$. To construct the transition probability matrix of the discrete-time Markov chain we define an $N \times N$ binary matrix

$\mathbf{G}$ such that

$$g_{ij} = \begin{cases} 1 & \text{if an arc exists from node } j \text{ to node } i \text{ in } \mathcal{G}, \\ 0 & \text{otherwise.} \end{cases}$$

It follows that

$$\mu_+ \Big(\sum_i g_{ij} \Big)^{-1}$$

is the probability of moving forward from node j along any of its outgoing arcs, and

$$\mu_- \Big(\sum_i g_{ji} \Big)^{-1}$$

is the probability of moving backward from node j along any of its incoming arcs. Therefore, the transition probability matrix is given by

$$\mathbf{B} = \mu_0 \mathbf{I} + \mu_+ \mathbf{G}\,\mathrm{diag}(\mathbf{1}^\top \mathbf{G})^{-1} + \mu_- \mathbf{G}^\top \mathrm{diag}(\mathbf{G}\mathbf{1})^{-1}.$$

We note that although $\mathbf{B}$ is nonsymmetric, it has symmetric nonzero structure.

For our numerical tests we set $\mu_0 = 0$, $\mu_- = 0.05$, and $\mu_+ = 0.95$, and let

$$(n_x, n_y) \in \{(32, 16), (64, 32), (128, 64), (256, 128), (256, 256)\}.$$

The results for the octagonal mesh problem are given in Table 3.6. We observe optimal performance that is independent of the problem size for both the MCAMG and MCAMG-hybrid methods. While the hybrid method requires twice the iterations to converge, it does so in only half the amount of time. Thus, one hybrid cycle is on average four times faster than an MCAMG cycle for this problem. The execution time scaling of the two methods is given in Figure 3.13.

Method	n	nnz	it	γ	time	C_{op}	C_{grid}	R_{lump}	levs
	4096	16192	20	0.28	0.7	3.73	1.95	0.010	8
	16384	65152	20	0.29	2.7	3.85	1.94	0.010	9
MCAMG	65536	261376	20	0.29	11.8	3.89	1.93	0.010	11
	262144	1047040	20	0.29	51.7	3.91	1.93	0.009	12
	524288	2095104	20	0.29	106.5	3.90	1.93	0.010	13
	4096	16192	40(0)	0.59	(0.2, 0.1)	3.73	1.95	0.011	8
	16384	65152	39(0)	0.59	(0.7, 0.5)	3.85	1.94	0.012	9
MCAMG-h	65536	261376	39(0)	0.59	(3.1, 2.2)	3.90	1.93	0.012	11
	262144	1047040	39(0)	0.59	(13.5, 9.7)	3.91	1.93	0.012	12
	524288	2095104	39(0)	0.59	(27.6, 20.0)	3.90	1.93	0.012	13

Table 3.6: Octagonal mesh. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. The bracketed times (t_0, t_1) are the setup and solve times in seconds. A bracketed value beside an iteration count is the number of additional multiplicative cycles performed during the solve phase.

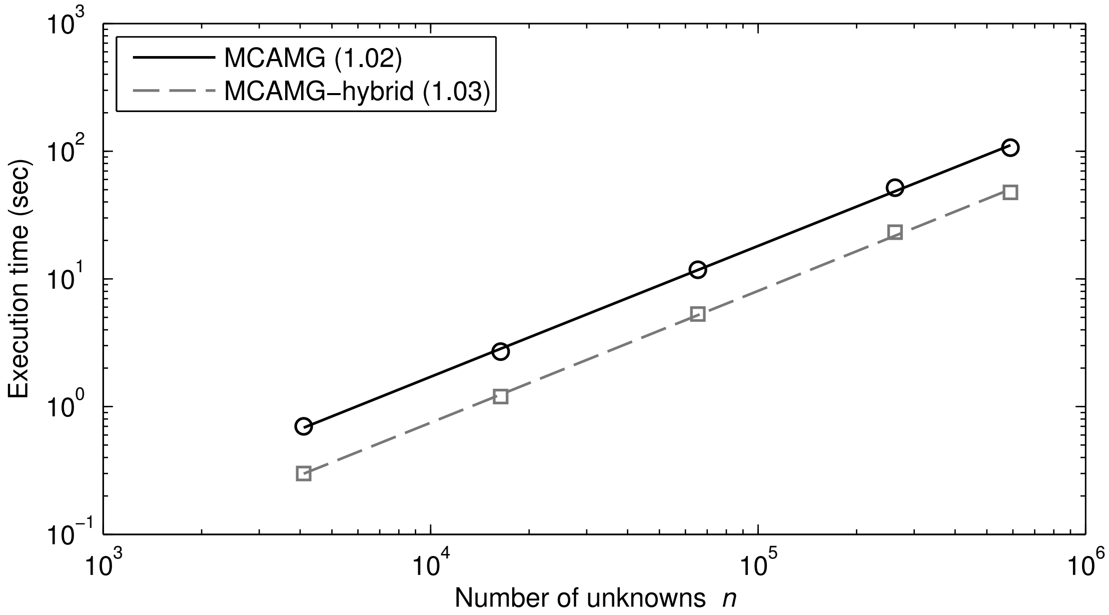


Figure 3.13: Octagonal mesh execution time scaling of MCAMG and MCAMG-hybrid. The solid and dashed lines are the best-fit lines through the data points (circles and squares). Numerical values in the legend are the slopes of the best-fit lines.

3.7.6 Stochastic Petri net

The next test problem we consider is derived from a stochastic Petri net (SPN) [6, 92]. Petri nets are a modeling formalism for the description of concurrency and synchronization in distributed systems. In general, Petri nets 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 input places and output places. An arc always runs from a place to a transition (input arc), or from a transition to a place (output arc). In a graphical representation of Petri nets it is customary to use circles to denote places, filled dots to denote tokens, rectangles to denote transitions, and arrowed lines to denote arcs. For example, the SPN in Figure 3.14 is taken from [92].

Tokens move between places according to *firing rules* imposed by the transitions. A transition can *fire* when it is *enabled*, that is, when each of its input places contains at least one token; when it fires, the transition consumes one token from each input place, and deposits one token in each of its output places. While multiple transitions may be enabled, it is assumed that only one transition fires at a time. Any distribution of tokens over the places represents a state of the model called a *marking*. Typically, a Petri is specified with an initial marking. Starting from the initial marking and following the firing rules we can progress through the states of the model. The *reachability set* of a Petri net is the set of all markings the net can reach, starting from an initial marking and following the firing rules. Moreover, the *reachability graph* of a Petri net is the directed graph in which each node corresponds to a reachable marking, and each arc (i, j) is labeled by the transition that fired to move the model from state i to state j.

Stochastic Petri nets are a way to add timing information into the Petri net modeling language. An SPN is a standard Petri net together with a tuple $(\lambda_1, \ldots, \lambda_n)$ of firing rates. Once transition t_i is enabled, there is an exponentially distributed delay time with rate λ_i until it can fire. As discussed in [92], a finite place, finite transition, stochastic Petri net with a specified initial marking is isomorphic to a one-dimensional Markov chain. To construct this Markov chain consider the reachability graph of the SPN and suppose

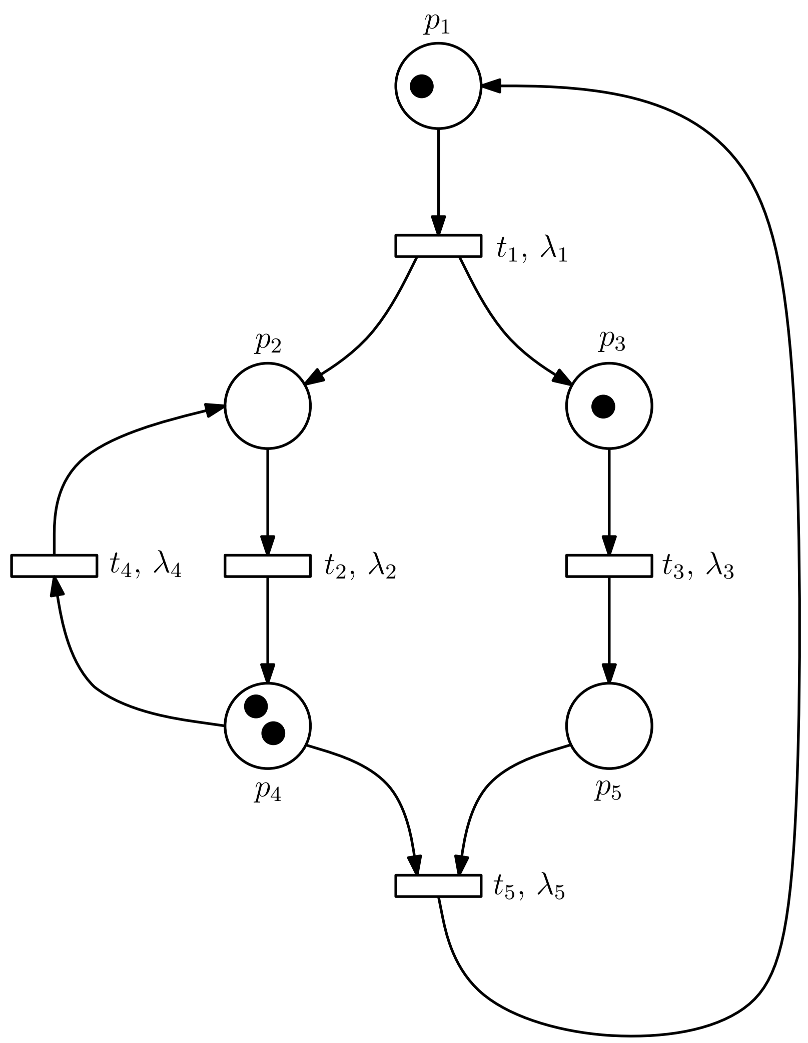


Figure 3.14: Graphical representation of a stochastic Petri net with initial marking $M_0 = (1, 0, 1, 2, 0)$. Places are labeled by $(p_1, p_2, p_3, p_4, p_5)$ and transitions are labeled by $(t_1, t_2, t_3, t_4, t_5)$ together with their corresponding firing rate $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5)$.

that each arc is weighted by the firing rate of its corresponding transition. The resulting weighted graph is equivalent to the transition diagram of a continuous-time Markov chain, which defines the infinitesimal generator matrix of the chain.

Test	Initial marking	n	nnz
1	$(22, 0, 0, 0, 0)$	4324	24058
2	$(35, 0, 0, 0, 0)$	16206	92646
3	$(55, 0, 0, 0, 0)$	60116	349636
4	$(90, 0, 0, 0, 0)$	255346	1502956
5	$(115, 0, 0, 0, 0)$	527046	3115006

Table 3.7: MCAMG test cases for stochastic Petri net.

Method	Test	it	γ	time	C_{op}	C_{grid}	R_{lump}	levs
MCAMG	1	18	0.24	0.6	2.42	2.12	0.001	8
	2	19	0.25	2.4	2.50	2.15	0.001	10
	3	26	0.41	15.1	2.55	2.16	0.001	10
	4	27	0.43	81.6	2.59	2.17	0.001	13
	5	27	0.43	165.1	2.60	2.17	0.001	14
MCAMG-h	1	19(9)	0.28	$(0.2,\ 0.3)$	2.42	2.12	0.001	8
	2	19(12)	0.27	$(0.9,\ 1.4)$	2.50	2.15	0.001	10
	3	37(4)	0.56	$(3.9,\ 4.1)$	2.65	2.17	0.003	11
	4	31(11)	0.49	$(19.8, 38.2)$	2.60	2.17	0.002	12
	5	30(8)	0.48	$(42.0, 65.2)$	2.82	2.18	0.005	14

Table 3.8: Stochastic Petri net. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. The bracketed times (t_0, t_1) are the setup and solve times in seconds. A bracketed value beside an iteration count is the number of additional multiplicative cycles performed during the solve phase, which is included in the overall iteration count.

As our test problem we consider the SPN described by Figure 3.14 with firing rates $(1, 3, 7, 9, 5)$. Numerical tests correspond to the initial markings given in Table 3.7. The results for the stochastic Petri net problem are given in Table 3.8. In order to obtain reasonable operator complexities it was necessary to use the strength threshold $\theta = 0.7$. The standalone MCAMG method displays near-optimal performance with small operator complexities and very acceptable execution time scaling (Figure 3.15). We observe that although the hybrid approach leads to improved execution times, the improvement is only moderate. The lackluster performance of MCAMG-hybrid is attributable to the large number of extra multiplicative cycles during the solve phase. The primarily reason for these extra multiplicative cycles was to correct for unacceptably large negative components in the additive iterate. For $n = 255346$ the extra multiplicative cycles account for 80% of the overall solve time, and for $n = 527046$ the extra multiplicative cycles account for 70% of the overall solve time!

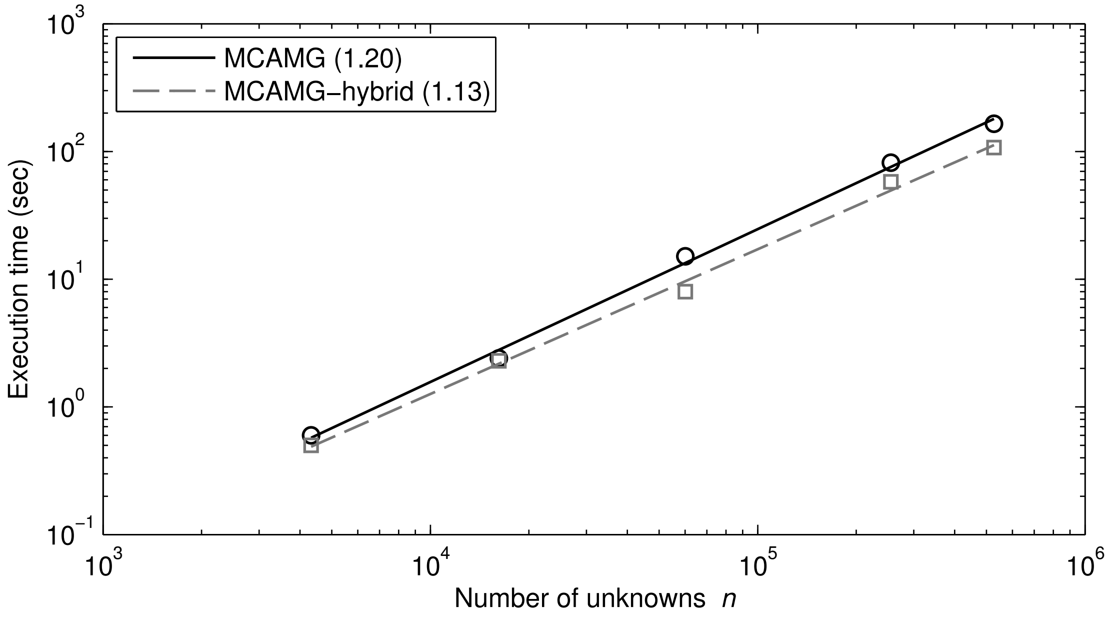


Figure 3.15: Stochastic Petri net execution time scaling of MCAMG and MCAMG-hybrid. The solid and dashed lines are the best-fit lines through the data points (circles and squares). Numerical values in the legend are the slopes of the best-fit lines.

3.7.7 MARCA ATM

The next problem we consider is an example of a multi-class, finite buffer priority system that has been considered in the telecommunications literature as a model for asynchronous transfer mode (ATM) networks [100]. The model consists of a single service center at which two identical servers provide service to two classes of customers. The service rates for each class μ_1 and μ_2 are exponentially distributed. An illustration of the model is given in Figure 3.16. For class 1 customers, ν_1 and ν_2 are the rates of the two phases in the arrival process, and p is the probability of taking the first of these. Similarly, for class 2 customers, γ_1 and γ_2 are the rates of the two phases in the arrival process, and q is the probability of taking the first of these. A six component vector is used to represent the states of the underlying Markov chain. Components 1 and 2 denote the phase of the arrival process for each of the two classes, respectively. Components 3 and 4 represent the number of class 1 and class 2 customers in the system. Components 5 and 6 indicate the state of the two servers. The number of states in the Markov chain can be increased by increasing

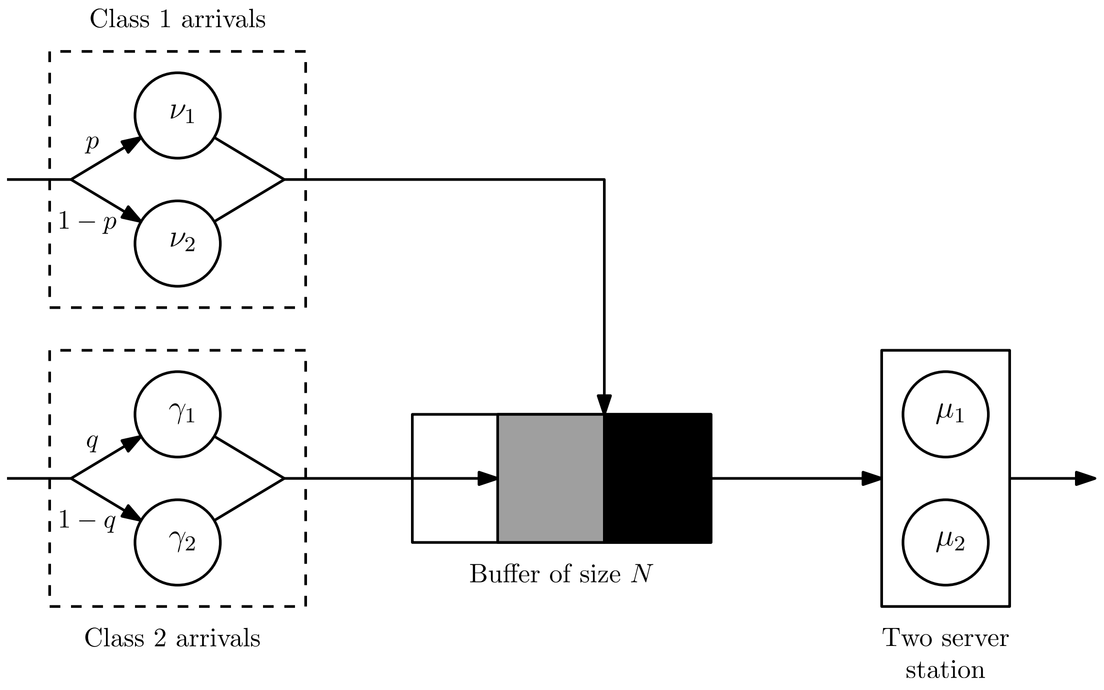


Figure 3.16: Illustration of the MARCA ATM model.

the buffer size N. We use the following parameters for our tests

$$(p, q, \nu_1, \nu_2, \gamma_1, \gamma_2, \mu_1, \mu_2) = (0.25, 0.5, 2.0, 3.0, 2.0, 3.0, 1.0, 1.5)$$

and consider buffer sizes of $N = 12$, 23, 46, 91, 181. The code and data files used to build the transition rate matrix for this Markov chain model are provided freely on the web [109].

Method	n	nnz	it	γ	time	C_{op}	C_{grid}	R_{lump}	levs
MCAMG	4068	27076	24	0.35	0.9	3.29	2.10	0.011	8
	16580	111164	24	0.35	3.4	2.97	2.05	0.007	10
	65540	440924	24	0.34	15.2	2.76	2.04	0.004	12
	260660	1756544	24	0.35	64.1	2.62	2.02	0.003	13
MCAMG-h	4068	27076	32(0)	0.50	(0.4, 0.1)	3.33	2.11	0.012	9
	16580	111164	31(2)	0.48	(1.0, 0.8)	2.96	2.05	0.007	10
	65540	440924	31(7)	0.46	(6.2, 5.3)	2.75	2.04	0.004	11
	260660	1756544	32(7)	0.47	(29.3, 28.9)	3.60	2.08	0.014	14

Table 3.9: MARCA ATM. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. The bracketed times (t_0, t_1) are the setup and solve times in seconds. A bracketed value beside an iteration count is the number of additional multiplicative cycles performed during the solve phase, which is included in the overall iteration count.

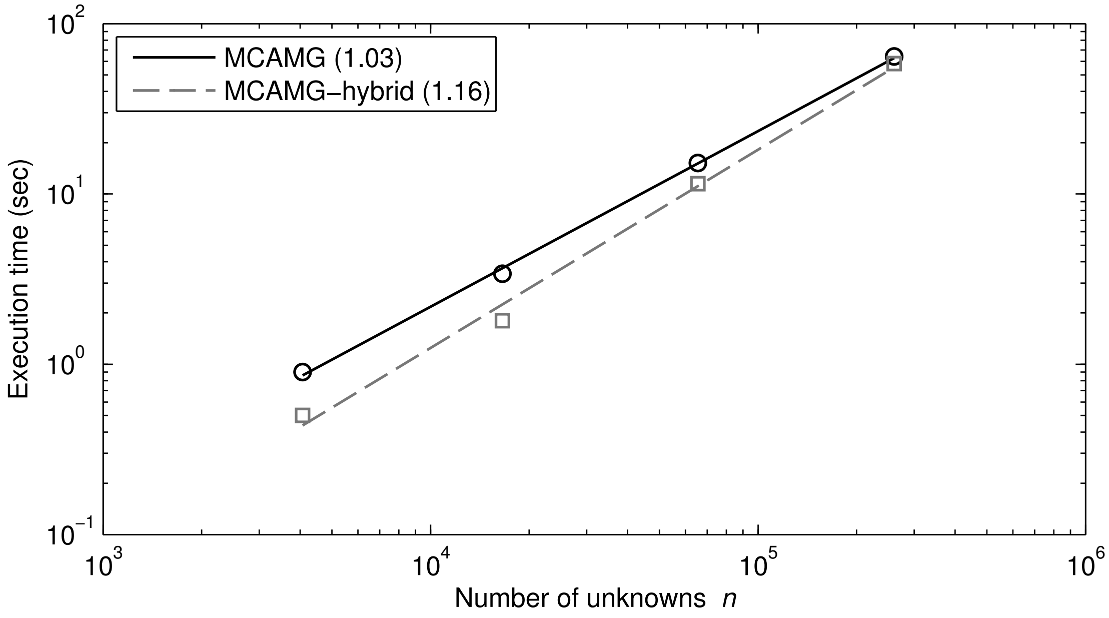


Figure 3.17: MARCA ATM problem execution time scaling of MCAMG and MCAMG-hybrid. The solid and dashed lines are the best-fit lines through the data points (circles and squares). Numerical values in the legend are the slopes of the best-fit lines.

The results for the MARCA ATM problem are given in Table 3.9. In order to obtain reasonable operator complexities it was necessary to use the larger strength threshold $\theta = 0.5$. We observe near-optimal performance of the MCAMG method, with very reasonable operator complexities and good convergence rates. For $n = 260660$ the seven extra multiplicative cycles during the solve phase of MCAMG-hybrid account for over half the solve time. Consequently, the hybrid approach is unable to significantly improve upon MCAMG. Execution time scaling of MCAMG and MCAMG-hybrid is given in Figure 3.17.

3.7.8 Reliability model

The final test problem we consider is a simple reliability model [8, 110] in which there are two different classes of machines, each subject to breakdown and a subsequent repair. It is assumed that each class has the same number of machines. The states of the system are represented by tuples (n_1, n_2), where n_i is the number of functioning machines in the ith class. Thus, if there are N machines per class, the model has $(N+1)^2$ states in total. The times between successive breakdowns and successive repairs are exponentially distributed. The breakdown rates of the class 1 and class 2 machines are λ_1 and λ_2, respectively. Similarly, the repair rates of the class 1 and class 2 machines are μ_1 and μ_2, respectively. The possible transitions and the rates at which they occur are given by

$$\begin{aligned}
(n_1, n_2) &\rightarrow (n_1+1, n_2) \quad \text{with rate} \quad \mu_1(N-n_1),\\
(n_1, n_2) &\rightarrow (n_1-1, n_2) \quad \text{with rate} \quad \lambda_1 n_1,\\
(n_1, n_2) &\rightarrow (n_1, n_2+1) \quad \text{with rate} \quad \mu_2(N-n_2),\\
(n_1, n_2) &\rightarrow (n_1, n_2-1) \quad \text{with rate} \quad \lambda_2 n_2,
\end{aligned}$$

for $0 \leq n_1, n_2 \leq N$. If we order the states such that state (i, j) has index $(N+1)(N-i) + N - j + 1$, then the resulting $(N+1)^2 \times (N+1)^2$ column-oriented infinitesimal generator matrix $\mathbf{Q}$ is block tridiagonal with stencil $\mathbf{Q} = [\mathbf{B}_k,\ \mathbf{A},\ \mathbf{C}_k]$, where

$$\mathbf{A} = \begin{bmatrix} * & \mu_2 & & & \\ N\lambda_2 & * & 2\mu_2 & & \\ & (N-1)\lambda_2 & * & \ddots & \\ & & \ddots & \ddots & N\mu_2 \\ & & & \lambda_2 & * \end{bmatrix}, \quad \mathbf{B}_k = (N-k+1)\lambda_1\mathbf{I}, \quad \text{and} \quad \mathbf{C}_k = k\mu_1\mathbf{I},$$

for $k = 1, \ldots, N$. The diagonal elements indicated by asterisks are the negated sums of the off-diagonal elements in their corresponding columns in $\mathbf{Q}$. In our numerical tests we take $\lambda_1 = 0.2$, $\lambda_2 = 30$, $\mu_1 = 0.5$, and $\mu_2 = 60$, which leads to a case of slow mixing.

Method	n	nnz	it	γ	time	C_{op}	C_{grid}	R_{lump}	levs
	4096	20224	15	0.25	0.4	2.41	2.00	0.000	9
	16384	81408	22	0.41	2.2	2.56	2.02	0.004	11
MCAMG	65536	326656	12	0.13	5.2	2.59	2.02	0.004	13
	262144	1308672	12	0.14	22.6	2.59	2.02	0.003	14
	589824	2946048	12	0.15	56.6	2.58	2.01	0.002	16
	4096	20224	16(12)	0.26	(0.1, 0.3)	2.40	1.99	0.000	8
	16384	81408	29(6)	0.54	(0.3, 0.9)	2.56	2.03	0.004	11
MCAMG-h	65536	326656	14(4)	0.19	(1.4, 2.4)	2.58	2.02	0.004	13
	262144	1308672	14(4)	0.20	(5.9, 10.5)	2.63	2.02	0.004	14
	589824	2946048	15(5)	0.25	(13.7, 29.6)	2.62	2.02	0.004	16

Table 3.10: Reliability model. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. The bracketed times (t_0, t_1) are the setup and solve times in seconds. A bracketed value beside an iteration count is the number of additional multiplicative cycles performed during the solve phase, which is included in the overall iteration count.

The test results for the reliability model are given in Table 3.10. We observe near-optimal performance of the MCAMG method, with very reasonable operator complexities and good convergence rates. We note that it is unclear what causes the spike in the convergence factors for $n = 16384$; however, the rate of convergence of MCAMG for $n = 16384$ can be improved substantially by using a V$(3,3)$-cycle as opposed to a V$(2,2)$-cycle.

In this case the hybrid method is unable results in only a moderate speedup of MCAMG, in part due to the relatively high number of additional multiplicative cycles during the additive solve phase. Execution time scaling of MCAMG and MCAMG-hybrid is given in Figure 3.18.

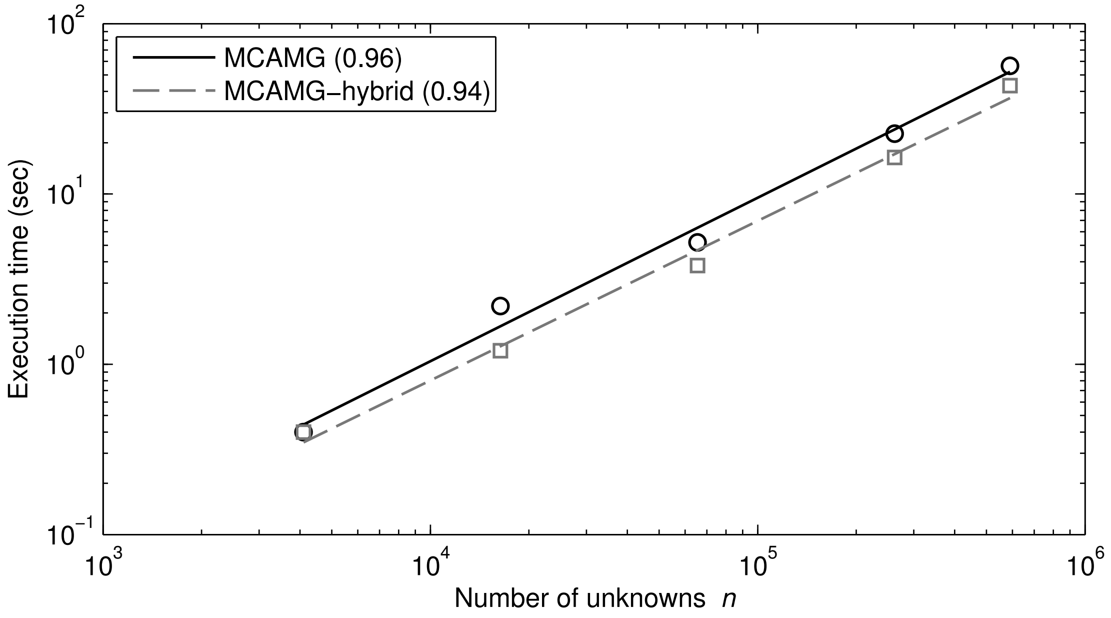


Figure 3.18: Reliability model execution time scaling of MCAMG and MCAMG-hybrid. The solid and dashed lines are the best-fit lines through the data points (circles and squares). Numerical values in the legend are the slopes of the best-fit lines.

3.8 General discussion and conclusions

The main contribution of this chapter was to show how classical AMG techniques can be applied in an exact interpolation scheme framework to compute the stationary distribution of irreducible homogeneous Markov chains. In particular, it was shown through a qualitative analysis that algebraically smooth multiplicative error is locally constant along strong connections in the scaled operator $\bar{\mathbf{A}}$, which motivated the use of the classical AMG coarsening and interpolation. The significance of this result is that it demonstrates how classical AMG coarsening and interpolation can be extended to a specific class of non-symmetric matrices, that is, irreducible singular M-matrices with zero column sums. The MCAMG algorithm was vetted through a series of challenging numerical test cases, for which it demonstrated near-optimal performance and scalability. Moreover, it was shown how a simple hybrid method with an MCAMG setup phase and an additive solve phase could provide significant speedups of standalone MCAMG in some cases. We conclude this chapter with a few general observations.

In AMG it is well-known that Galerkin coarse-level operators tend to lose sparsity on the coarser levels which can lead to large operator complexities. Our numerical tests revealed that the operator complexity of MCAMG V-cycles is somewhat sensitive to the strength threshold θ. Furthermore, for the stochastic Petri net, MARCA ATM model, and MARCA NCD model, operator complexities were substantially larger during the initial iterations, but then tended to settle down at a decreased level as solution accuracy improved. It is difficult to say exactly why the operator complexity may initially be large for some problems and not for others; however, this phenomenon appears to depend on how close the coarse-level iterates are to the coarse vector of all ones $\mathbf{1}_c$. Since we base strength of connection on the scaled system operator (which is motivated by the assumption of small residuals; see §3.1), it is plausible that operator complexities may initially increase when coarse-level iterates are far from $\mathbf{1}_c$, and then stabilize at some level as the coarse-level iterates approach $\mathbf{1}_c$. Often, simply increasing θ was sufficient to reduce the operator complexity to an acceptable value. Therefore, it may be worthwhile to investigate an iteration-dependent strength threshold that adapts with respect to the measured operator complexity. However, this work is beyond the scope of this thesis. As a compromise we advocate a larger strength threshold. In general, $\theta = 0.5$ seems to work well, however, in certain rare cases it may be necessary to choose $\theta \geq 0.9$.

While the stationary distribution vector of an irreducible Markov chain is guaranteed to have strictly positive components by the Perron–Frobenius theorem, for certain Markov chains the majority of these components can be extremely close to zero, on the order of 10^{-300} or smaller. Moreover, the stationary distribution may consist primarily of small values with a few probabilities that are $\mathcal{O}(1)$. Owing to the nature of the multiplicative cycles, this discrepancy in scales may lead to poorly scaled coarse-level system operators in which some diagonal entries are essentially equal to zero. As a consequence, weighted Jacobi relaxation may "blow up" due to overflow errors caused by numerical roundoff. In this situation Kaczmarz relaxation [104, 115] on the coarser levels, which is often employed in multigrid when the coarser-level systems are ill-conditioned, was also ineffective. In addition, extremely small values in the computed solution may also lead to spurious diagonal and off-diagonal values in the lumped coarse-level matrices. For example, these issues arose with the reliability model test problem. The simplest and most effective remedy we have found for these issue is as follows. The pre-relaxed iterate is examined for any extremely small values by comparing each of its components with a small positive threshold ε. Any components found below this threshold are set to ε. To limit the amount of additional work, this procedure can be interwoven into the updates of the final iteration of the relaxation method. Moreover, because the relaxed iterates on the coarser levels tend to $\mathbf{1}_c$ after only a few MCAMG cycles, it should only be necessary to perform this check on the finest level. After some experimentation, a reasonable value for ε seems to be 10^{-50}. We note that an approach similar to ours was advocated in [110], albeit for an unrelated method, by setting components of the computed solution below a certain threshold to zero.

Negative values of significant magnitude may occur in the early stages of the additive solve phase when the solution is still far from the exact solution. When designing the hybrid method it was unclear whether these negative values should be ignored, or addressed as in steps 5 and 6 of Algorithm 3.2. Numerical experiments in which negative values were ignored by setting $\delta = \infty$ in Algorithm 3.2 converged poorly with a large number of iterations in the solve phase. Thus, it appears that the extra computational work of performing additional multiplicative cycles to obtain strictly positive iterates in the additive solve phase is justified and necessary.

Chapter 4

Top-Level Acceleration of AMG Methods for Markov Chains

In this chapter we discuss a simple method based on *iterant recombination* [119] to accelerate multigrid methods for computing the stationary distribution of Markov chains. While our approach can be applied to any multilevel Markov chain algorithm that satisfies a certain minimal set of assumptions, we limit our scope to the non-overlapping adaptive multilevel aggregation algorithm developed in [51], which is closely related to the earlier work of Horton and Leutenegger [69]. In general, basic multilevel aggregation schemes with non-overlapping aggregates rarely achieve algorithmic scalability due to the inability of the aggregation-based transfer operators to accurately represent near-nullspace components of the fine-level system operator in their range. Thus, basic multilevel aggregation is a prime candidate to demonstrate the improvements that may be gained by our acceleration approach.

Iterant recombination constructs an improved fine-level approximation as a linear combination of successive fine-level iterates from previous multigrid cycles, where the linear combination minimizes the residual with respect to some norm. In this respect multigrid acceleration by iterant recombination is closely related to multigrid-preconditioned Krylov subspace iterations. For example, restarted GMRES with multigrid preconditioning is theoretically equivalent to multigrid acceleration by iterant recombination with a fixed number of previous iterates and two-norm residual minimization [119]. Consequently, the distinction between multigrid as a preconditioner and multigrid accelerated by iterant recombination depends largely on one's perspective. Since the multiplicative schemes we consider have multigrid hierarchies that evolve with each iteration, standard Krylov acceleration is not applicable because the spaces involved are not related by a fixed multigrid preconditioner. Moreover, since the iterant recombination process is formulated as a constrained minimization problem over a subspace of probability vectors, flexible acceleration techniques such as FGMRES [104] are not readily applicable. Instead, the resulting minimization problem is solved by techniques from constrained optimization. We note that GMRES preconditioned by additive AMG with a *fixed* hierarchy has already been considered in the literature for Markov chains, for example, see [127].

In what follows we consider minimizing the two-norm and the one-norm of the residual with nonnegativity constraints. Minimization in the two-norm results in a quadratic programming problem that is solved using standard techniques from quadratic optimization. Minimization in the one-norm results in a nonlinear convex programming problem that we solve by a variant of the ellipsoid method. We consider the ellipsoid method because it is straightforward to implement efficiently, it is a robust solver for nonlinear programming problems [57], and at some levels of solution error it is competitive with other general-purpose solvers [57]. We consider minimizing in the one-norm primarily to determine if faster overall acceleration can be obtained compared with minimizing in the two-norm. Furthermore, there are two additional but less significant motivations for considering minimization in the one-norm. In probability theory the one-norm is used to measure distances between probability vectors, which is natural since probability vectors are unit vectors in the one-norm. It is then only natural to consider minimization in the one-norm. Also, one-norm minimization methods have recently raised significant interest in emerging fields such as compressive sensing, sparse representation, and sparse factorization. Consequently the question of whether one-norm minimization can be done efficiently as compared to two-norm minimization is receiving greater attention.

Acceleration of multigrid methods by iterant recombination dates back at least as far as the paper by Brant and Mikulinsky [25]. Accelerators similar to ours have been designed for other nonlinear iterations, for example, Washio and Oosterlee [128] develop an accelerator for the full approximation scheme (FAS) (see §2.6.5 for a brief overview) in the context of solving nonlinear partial differential equations. A key difference between their approach and ours is that our acceleration method for Markov chains 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, whereas our multiplicative approach does not rely on linearization. Nevertheless, the FAS accelerator does share many characteristics of the method developed here.

The rest of this chapter is organized as follows. We begin by describing the adaptive multilevel aggregation algorithm for Markov chains. In §4.2 we discuss the constrained iterant recombination approach. In §4.3 we discuss Matlab's built-in quadratic programming solver `quadprog` as a solver for the two-norm iterant recombination optimization problem. We also describe an efficient algorithm for computing the analytical solution of the quadratic programming problem with window size two. Sections 4.4 and 4.5 describe the ellipsoid method for nonlinear convex programs and discuss how it can be applied to solve the one-norm optimization problem arising from the iterant recombination process. Section 4.6 briefly discusses the connection between the one-norm minimization problem and linear programming. In §4.7 we present the numerical results, and §4.8 contains the concluding remarks.

4.1 Multilevel aggregation for Markov chains

We begin by describing the adaptive multilevel aggregation method for Markov chains (AGG) that we aim to accelerate. Our description is brief since the underlying multiplicative aggregation framework is essentially the same as the framework used by the MCAMG method (see [51] for further details). As before we use two-level notation in which coarse-level quantities are denoted by a subscript "c", while fine-level quantities and transfer operators do not carry any subscripts. The fine-level problem we wish to solve is given by

$$\mathbf{A}\mathbf{x} = \mathbf{0}, \qquad x_i > 0 \ \text{ for } \ i = 1, \ldots, n, \qquad \sum_{i=1}^{n} x_i = 1, \tag{4.1}$$

where $\mathbf{A} = \mathbf{I} - \mathbf{B}$ is an irreducible singular M-matrix with zero column sums. Decomposing the fine-level degrees of freedom $\{1, \ldots, n\}$ into n_c mutually disjoint aggregates $\mathcal{A}_1, \ldots, \mathcal{A}_{n_c}$, the $n \times n_c$ full rank disaggregation operator $\mathbf{Q}$ is defined by

$$q_{ij} = \begin{cases} 1 & \text{if } i \in \mathcal{A}_j, \\ 0 & \text{otherwise.} \end{cases}$$

The aggregated coarse-level problem is then defined by the classical aggregation equations proposed by Simon and Ando [107], that is,

$$\mathbf{A}_c \mathbf{x}_c = \mathbf{0}_c, \tag{4.2}$$

where

$$\mathbf{A}_c = \mathbf{Q}^\top \operatorname{diag}(\bar{\mathbf{x}}^{(k)}) \mathbf{Q} \operatorname{diag}(\mathbf{Q}^\top \bar{\mathbf{x}}^{(k)})^{-1} - \mathbf{Q}^\top \mathbf{B} \operatorname{diag}(\bar{\mathbf{x}}^{(k)}) \mathbf{Q} \operatorname{diag}(\mathbf{Q}^\top \bar{\mathbf{x}}^{(k)})^{-1}, \tag{4.3}$$

with $\bar{\mathbf{x}}^{(k)}$ the relaxed fine-level iterate. We assume that $\bar{\mathbf{x}}^{(k)}$ has strictly positive elements for all iterations and on all levels, which can be proved by arguments similar to those found in §3.5. Defining the aggregation-based full rank restriction and interpolation operators by

$$\mathbf{R} = \mathbf{Q}^\top \quad \text{and} \quad \mathbf{P} = \operatorname{diag}(\bar{\mathbf{x}}^{(k)}) \mathbf{Q} \operatorname{diag}(\mathbf{Q}^\top \bar{\mathbf{x}}^{(k)})^{-1}, \tag{4.4}$$

the equation for the coarse-level system operator can be written more succinctly as

$$\mathbf{A}_c = \mathbf{R}\mathbf{A}\mathbf{P}. \tag{4.5}$$

Given the definition of $\mathbf{A}_c$ in (4.3), the coarse-level system (4.2) can be interpreted as a coarse-level probability equation, as we now show. Defining the coarse-level stochastic matrix $\mathbf{B}_c$ by

$$\mathbf{B}_c \coloneqq \mathbf{Q}^\top \mathbf{B}\,\mathrm{diag}(\bar{\mathbf{x}}^{(k)})\mathbf{Q}\,\mathrm{diag}(\mathbf{Q}^\top \bar{\mathbf{x}}^{(k)})^{-1}, \tag{4.6}$$

we find that

$$\begin{aligned}\mathbf{A}_c &= \mathbf{R}(\mathbf{I} - \mathbf{B})\mathbf{P}\\ &= \mathbf{Q}^\top \mathrm{diag}(\bar{\mathbf{x}}^{(k)})\mathbf{Q}\,\mathrm{diag}(\mathbf{Q}^\top \bar{\mathbf{x}}^{(k)})^{-1} - \mathbf{Q}^\top \mathbf{B}\,\mathrm{diag}(\bar{\mathbf{x}}^{(k)})\mathbf{Q}\,\mathrm{diag}(\mathbf{Q}^\top \bar{\mathbf{x}}^{(k)})^{-1}\\ &= \mathbf{I}_c - \mathbf{B}_c.\end{aligned}$$

Consequently, by Definition 2.2.2 the coarse-level system operator $\mathbf{A}_c$ is a singular M-matrix. Furthermore, we can prove that $\mathbf{A}_c$ is irreducible by arguments identical to those found in the proof of Proposition 3.5.1. As a consequence, we do not require any lumping to obtain an irreducible singular M-matrix on the coarse level. The coarse-level problem (4.2) also has the following straightforward probabilistic interpretation. If $\bar{\mathbf{x}}^{(k)}$ is equal to the exact fine-level solution $\mathbf{x}$, then the solution of (4.2) is given by

$$\mathbf{x}_c = \mathbf{Q}^\top \mathbf{x} \quad \Leftrightarrow \quad (\mathbf{x}_c)_i = \sum_{j \in \mathcal{A}_i} x_j \;\text{ for }\; i = 1, \ldots, n_c.$$

That is, the solution of the coarse-level aggregated system truly represents an aggregated version of the exact fine-level solution. The coarse-grid correction corresponding to the two-level method is given by

$$\mathbf{x}^{\mathrm{CGC}} = \mathbf{P}\mathbf{x}_c. \tag{4.7}$$

We note that if $\bar{\mathbf{x}}^{(k)} = \mathbf{x}$, then $\mathbf{x}$ is a fixed point of the two-level method:

$$\mathbf{x}^{\mathrm{CGC}} = \mathbf{P}\mathbf{x}_c = \mathrm{diag}(\mathbf{x})\mathbf{Q}\,\mathrm{diag}(\mathbf{Q}^\top \mathbf{x})^{-1}\mathbf{Q}^\top \mathbf{x} = \mathbf{x}. \tag{4.8}$$

A pseudocode description of the multilevel AGG method is given by Algorithm 4.1. We introduce a new multigrid cycle in Algorithm 4.1, namely the F-cycle, which is an intermediate case between a V-cycle and W-cycle. F-cycles are often used in practice because their computational cost is similar to a V-cycle and their convergence factor is similar to a W-cycle. A common strategy to improve the convergence of multilevel aggregation schemes is to use either F-cycles or W-cycles. We note that F-cycles and W-cycles can result in large operator complexities in the case of overlapping aggregates, for example, in the case of the MCAMG method. However, operator complexities tend to remain small for multilevel aggregation schemes with non-overlapping aggregates (see §2.6.4). In Algorithm 4.1 if $\mu = 1$ we obtain a V-cycle, if $\mu = 2$ we obtain a W-cycle, and if $\mu = 3$ we obtain an F-cycle. Relaxations correspond to the weighted Jacobi method with $\omega \in (0, 1)$ and for the direct solver on the coarsest level we use the GTH algorithm (see §3.2).

Algorithm 4.1: AGG algorithm for Markov chains

```
   Input: $\mathbf{A}_\ell$, current approximation $\mathbf{x}_\ell^{(k)}$ cycle index $\mu$, smoothing steps $\nu_1$, $\nu_2$
   Output: New approximation $\mathbf{x}_\ell^{(k+1)}$
   if on the coarsest level then
1.     Solve $\mathbf{A}_\ell \mathbf{x}_\ell = \mathbf{0}_\ell$ subject to $\mathbf{x}_\ell$ strictly positive
   else
2.     Perform $\nu_1$ relaxations: $\bar{\mathbf{x}}_\ell^{(k)} \leftarrow \text{Relax}(\mathbf{A}_\ell,\, \mathbf{0}_\ell,\, \mathbf{x}_\ell^{(k)},\, \nu_1)$
3.     Compute the transfer operators $\mathbf{R}$ and $\mathbf{P}$ according to (4.4)
4.     Construct the coarse-level system operator $\mathbf{A}_{\ell+1} \leftarrow \mathbf{R}\mathbf{A}_\ell\mathbf{P}$
       if $\mu = 1$ then
5.         $\mathbf{x}_{\ell+1} \leftarrow \text{AGG}(\mathbf{A}_{\ell+1},\, \mathbf{1}_{\ell+1},\, \mu,\, \nu_1,\, \nu_2)$
       else
6.         $\mathbf{x}_{\ell+1} \leftarrow \text{AGG}(\mathbf{A}_{\ell+1},\, \mathbf{1}_{\ell+1},\, \mu,\, \nu_1,\, \nu_2)$
7.         $\mathbf{x}_{\ell+1} \leftarrow \text{AGG}(\mathbf{A}_{\ell+1},\, \mathbf{x}_{\ell+1},\, 3 - \lceil \mu/2 \rceil,\, \nu_1,\, \nu_2)$
       end
8.     Correct: $\mathbf{x}_\ell^{\text{CGC}} \leftarrow \mathbf{P}\mathbf{x}_{\ell+1}$
9.     Perform $\nu_2$ relaxations: $\mathbf{x}_\ell^{(k+1)} \leftarrow \text{Relax}(\mathbf{A}_\ell,\, \mathbf{0}_\ell,\, \mathbf{x}_\ell^{\text{CGC}},\, \nu_2)$
   end
```

We conclude this section by describing how the aggregation procedure. Aggregates are determined by the neighborhood-based aggregation technique of [123] with a symmetric strength of connection measure based on the scaled matrix

$$\bar{\mathbf{A}} = \mathbf{A}\,\mathrm{diag}(\bar{\mathbf{x}}^{(k)}).$$

The motivation behind basing strength of connection on $\bar{\mathbf{A}}$ is the same as in §3.1. Algebraically smooth multiplicative error varies slowly along strong connections in $\bar{\mathbf{A}}$, therefore, aggregates should contain strongly connected points. Given the strength of connection parameter $\theta \in (0, 1)$, a point i is strongly connected to a point j if either

$$-\bar{a}_{ij} \geq \theta \max_{k \neq i}\{-\bar{a}_{ik}\} \quad \text{or} \quad -\bar{a}_{ji} \geq \theta \max_{k \neq j}\{-\bar{a}_{jk}\}. \tag{4.9}$$

That is, points i and j are strongly connected if i is strongly influenced by j or if j is strongly influenced by i. Neighborhood aggregation relies on the notion of a *strong neighborhood* about a point i, denoted by $\mathcal{N}_i^s$, that is defined as the set all the points strongly connected to i with respect to a given strength of connection parameter θ, including i itself. We note that neighborhood-based aggregation is related to standard aggregation techniques in the AMG literature, but differs from the aggregation techniques for Markov chains given in [49, 51, 117]. Neighborhood-based aggregation is attractive for sparse problems with local connectivity because it typically results in well-balanced aggregates of approximately equal size, and in coarsening that reduces the number of unknowns quickly. In general, coarse-level stencil sizes tend to be uniform and do not grow quickly. The neighborhood aggregation scheme is given by Algorithm 4.2. The first pass of Algorithm 4.2 produces a set of tentative aggregates $\tilde{\mathcal{A}}_1, \ldots, \tilde{\mathcal{A}}_K$ corresponding to strong neighborhoods of the fine-level degrees of freedom. In the second pass each unassigned point i is added to the aggregate whose corresponding tentative aggregate has the most points in common with $\mathcal{N}_i^s$, that is, the most points to which i is strongly connected. We note that ties that arise in selecting j on line 6 of Algorithm 4.2 are broken arbitrarily.

Algorithm 4.2: Neighborhood aggregation

```
   Input: Strength of connection parameter $\theta$, $\bar{\mathbf{A}}$
   Output: Aggregates $\mathcal{A}_1, \ldots, \mathcal{A}_K$
1. Set $\mathcal{U} \leftarrow \{1, \ldots, n\}$ and $K \leftarrow 0$
   for $i \in \{1, \ldots, n\}$ do
2.     Let $\mathcal{N}_i^s$ be the strong neighborhood of $i$
       if $\mathcal{N}_i^s \subset \mathcal{U}$ then
3.         $K \leftarrow K + 1$
4.         Set $\mathcal{A}_K \leftarrow \mathcal{N}_i^s$ and $\tilde{\mathcal{A}}_K \leftarrow \mathcal{N}_i^s$
5.         $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{N}_i^s$
       end
   end
   while $\mathcal{U} \neq \emptyset$ do
6.     Select $i \in \mathcal{U}$ and set $j \leftarrow \text{argmax}_{k=1,\ldots,K} |\mathcal{N}_i^s \cap \tilde{\mathcal{A}}_k|$
7.     Set $\mathcal{A}_j \leftarrow \mathcal{A}_j \cup \{i\}$ and $\mathcal{U} \leftarrow \mathcal{U} \setminus \{i\}$
   end
```

4.2 Constrained iterant recombination

Suppose we have a sequence of successive iterates $\{\mathbf{x}^{(i)}\}_{i=1}^{k}$ on the finest level from previous multigrid cycles. In order to find an improved iterate $\mathbf{x}^\star$, we consider a linear combination of the m most recent iterates $\mathbf{x}^{(k)}, \mathbf{x}^{(k-1)}, \ldots, \mathbf{x}^{(k-m+1)}$, where m is the *window size*. Let $\mathbf{X}$ be the $n \times m$ matrix

$$\mathbf{X} = [\mathbf{x}^{(k-m+1)}, \ldots, \mathbf{x}^{(k-1)}, \mathbf{x}^{(k)}],$$

where $\mathbf{x}^{(k)}$ is the most recent iterate, and assume that each column of $\mathbf{X}$ is a probability distribution with strictly positive entries. We note that our assumption on the columns of $\mathbf{X}$ is met by the multilevel methods considered in this thesis. Then the improved approximation is given by $\mathbf{x}^\star = \mathbf{X}\mathbf{z}^\star$ for some vector $\mathbf{z}^\star \in \mathbb{R}^m$. This process is repeated after each multilevel cycle with $\mathbf{x}^\star$ serving as the initial guess for the next cycle; see Figure 4.1. We note that in practice iterant recombination can be applied in the first few multigrid cycles, in which case $\mathbf{X}$ may initially have fewer than m columns.

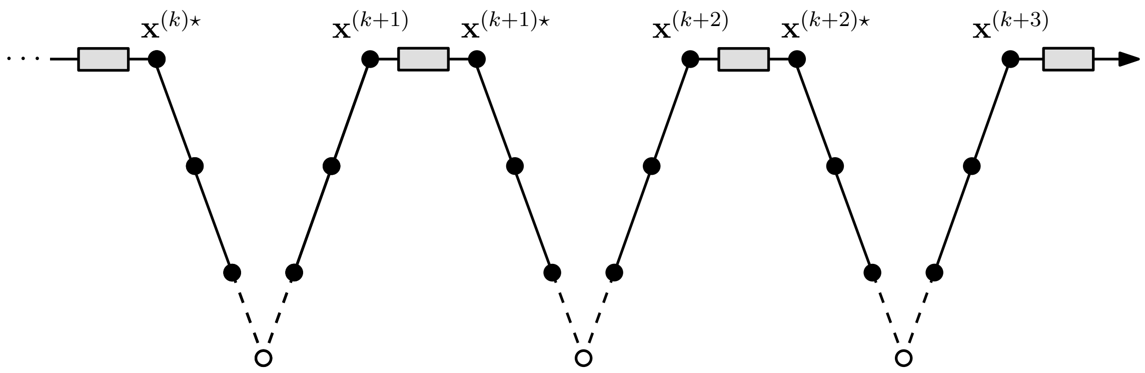


Figure 4.1: Accelerated multigrid V-cycles. The black dots (•) represent relaxation operations on their respective levels and the open dots (∘) represent coarse-level solves. An acceleration step, represented by a grey box, occurs after each V-cycle.

We require a criteria on which to base our choice of $\mathbf{z}^\star$, and hence $\mathbf{x}^\star$. Defining $\mathcal{P}$ as the set of all n-dimensional probability vectors

$$\mathcal{P} = \{\mathbf{u} \in \mathbb{R}^n : \|\mathbf{u}\|_1 = 1, u_i \geq 0 \text{ for } i = 1, \ldots, n\},$$

we choose a functional $F : \mathbb{R}^n \to \mathbb{R}$ such that the solution of $\mathbf{Ax} = \mathbf{0}$ is the unique global minimizer of F over $\mathcal{P}$. An obvious choice for F that satisfies this condition is $F(\mathbf{u}) = \|\mathbf{Au}\|$ where $\|\cdot\|$ is any vector norm on $\mathbb{R}^n$. Since the improved approximation belongs to $\mathcal{P}$ as well as to the range of $\mathbf{X}$, we define $\mathbf{x}^\star$ as the solution of the following minimization problem:

$$\text{minimize } F(\mathbf{u}) \text{ over } \mathcal{V} = \mathcal{P} \cap \operatorname{range}(\mathbf{X}), \tag{4.10}$$

where the set $\mathcal{V}$ is referred to as the *feasible set.* Since any vector belonging to the range of $\mathbf{X}$ can be written as $\mathbf{Xz}$, we have that $\mathbf{u} \in \mathcal{V}$ if and only if there exists some vector $\mathbf{z}$ such that

$$\mathbf{u} = \mathbf{Xz}, \quad (\mathbf{Xz})_i \geq 0 \text{ for } i = 1, \ldots, n, \quad \mathbf{1}^\top \mathbf{z} = 1. \tag{4.11}$$

Thus, (4.10) is equivalent to the following constrained minimization problem:

$$\begin{aligned} \text{minimize} \quad & F(\mathbf{Xz}) \\ \text{subject to} \quad & \mathbf{Xz} \geq \mathbf{0} \\ & \mathbf{1}^\top \mathbf{z} = 1 \\ & \mathbf{z} \in \mathbb{R}^m. \end{aligned} \tag{4.12}$$

The inequality constraints are necessary to maintain nonnegative signs throughout the computations, not only because nonnegative signs are desired for probability vectors, but also because our multilevel cycles may become ill-posed if iterates with negative signs occur. Since our multilevel cycles require that iterates have *strictly* positive components, we propose the following *tightened* minimization problem:

$$\begin{aligned} \text{minimize} \quad & F(\mathbf{Xz}) \\ \text{subject to} \quad & \mathbf{Xz} \geq \delta x_{min} \mathbf{1} \\ & \mathbf{1}^\top \mathbf{z} = 1 \\ & \mathbf{z} \in \mathbb{R}^m, \end{aligned} \tag{4.13}$$

where the parameter $\delta \in [0, 1)$ and x_{min} is the smallest element in $\mathbf{X}$. The lower bound δx_{min} ensures the new feasible set $\mathcal{V}_\delta \subset \mathcal{V}$ is not empty because any canonical basis vector in $\mathbb{R}^m$ belongs to $\mathcal{V}_\delta$. The modified inequality constraint has the added benefit of reducing the likelihood of any small negative values occurring in the computed solution. In practice we find that $\delta = 0.1$ works well. We note that for $\delta = 0$ the tightened minimization problem (4.13) is equivalent to the original minimization problem (4.12). Since the feasible set $\mathcal{V}_\delta$ is closed and convex with a nonempty interior, convexity of F implies that (4.13) is a convex program. Therefore, by a standard result from convex analysis, any local minimum of F on $\mathcal{V}_\delta$ is also a global minimum. In general the feasible set may be unbounded, and hence not compact, so it is difficult to say if there exists a solution of (4.13). In the special case that $F(\mathbf{u}) = \|\mathbf{Au}\|_2$, (4.13) is a quadratic programming problem. When $F(\mathbf{u}) = \|\mathbf{Au}\|_1$ the functional F is not C^1 and subgradient calculus is required for the ellipsoid algorithm.

There are m variables and $n \gg m$ inequality constraints in (4.13). In general, it is not possible to reduce the number of inequality constraints as they may all contribute to defining the set of feasible points. On the other hand, because of the single equality constraint, any vector belonging to the feasible set has only $m-1$ degrees of freedom. Thus, we can obtain an equivalent inequality-form problem with $m-1$ unknowns and n inequality constraints by eliminating one of the variables from (4.13). Eliminating z_1 and letting $\hat{\mathbf{z}} = (z_2, \ldots, z_m)^\top$, the equality constraint implies that $\mathbf{z} = (1 - \mathbf{1}^\top \hat{\mathbf{z}},\, \hat{\mathbf{z}})^\top$. Defining

$$\hat{\mathbf{X}} := -[\mathbf{x}_2, \ldots, \mathbf{x}_m] + \mathbf{x}_1 \mathbf{1}^\top, \quad \hat{\mathbf{A}} := -\mathbf{A}\hat{\mathbf{X}}, \quad \text{and} \quad \mathbf{r}_1 := \mathbf{A}\mathbf{x}_1, \tag{4.14}$$

where $\mathbf{x}_i$ is the ith column of $\mathbf{X}$, the inequality-form problem is given by

$$\begin{aligned} \text{minimize} \quad & \|\hat{\mathbf{A}}\hat{\mathbf{z}} + \mathbf{r}_1\| \\ \text{subject to} \quad & \hat{\mathbf{X}}\hat{\mathbf{z}} - \mathbf{x}_1 \leq -\delta x_{min}\mathbf{1} \\ & \hat{\mathbf{z}} \in \mathbb{R}^{m-1}. \end{aligned} \tag{4.15}$$

If $\hat{\mathbf{z}}^\star$ is the solution of (4.15), the improved iterate is given by $\mathbf{x}^\star = \mathbf{x}_1 - \hat{\mathbf{X}}\hat{\mathbf{z}}^\star$, and its residual is $\mathbf{r}_1 + \hat{\mathbf{A}}\hat{\mathbf{z}}^\star$. As we shall see, the inequality-form problem is necessary in practice when using the ellipsoid method to solve the one-norm minimization problem. We conclude this section with a pseudocode description of the iterant recombination procedure given by Algorithm 4.3. Due to roundoff errors the accelerated iterate $\mathbf{x}^{(k)\star}$ may have small negative components or zero components. If all negative components are sufficiently small in magnitude then $\mathbf{x}^{(k)\star}$ is overwritten by its absolute value (see (3.31) in §3.6). Otherwise, the accelerated iterate is rejected and the most recent multigrid iterate is used as the initial guess for the next multigrid cycle. Any zero components in $\mathbf{x}^{(k)\star}$ are replaced by the minimum between machine epsilon and the smallest positive component in $\mathbf{x}^{(k)\star}$. On line 7 of Algorithm 4.3 the improved iterate is rejected if it does not yield a strictly smaller residual than the residual of the most recent multigrid iterate, which may occur because (4.13) is solved only approximately. In addition to solving a minimization problem each iteration, the main computational overhead of Algorithm 4.3 is the computation of $\mathbf{X}\mathbf{z}^\star$ and the residuals $\mathbf{r}^{(k)}$ and $\mathbf{A}\mathbf{x}^{(k)\star}$, which requires $4\,\mathrm{nnz}(\mathbf{A}) + 2mn$ flops.

Algorithm 4.3: Iterant recombination with window size m

```
    Input: Initial guess $\mathbf{x}^{(0)}$, $\mathbf{A}$, window size $m$, convergence tolerance $\tau$
    Output: The converged approximation $\mathbf{x}^{(k)\star}$
1. Set $k \leftarrow 1$, $\tau_{rel} \leftarrow \tau\|\mathbf{A}\mathbf{x}^{(0)}\|_1$, and $\mathbf{x}^{(0)\star} \leftarrow \mathbf{x}^{(0)}$
2. Obtain the next multigrid iterate $\mathbf{x}^{(k)}$, with $\mathbf{x}^{(k-1)\star}$ as the initial guess
3. Set $j \leftarrow \min\{k, m\}$
4. Set $\mathbf{X} \leftarrow [\mathbf{x}^{(k-j+1)}, \ldots, \mathbf{x}^{(k-1)}, \mathbf{x}^{(k)}]$
5. Set $\mathbf{r}^{(k)} \leftarrow \mathbf{A}\mathbf{x}^{(k)}$ and update the matrix $\mathbf{AX}$
6. Solve (4.13) for $\mathbf{z}^\star$, and set $\mathbf{x}^{(k)\star} \leftarrow \mathbf{X}\mathbf{z}^\star$
7. if $\|\mathbf{A}\mathbf{x}^{(k)\star}\|_1 \geq \|\mathbf{r}^{(k)}\|_1$ then
       $\mathbf{x}^{(k)\star} \leftarrow \mathbf{x}^{(k)}$
   end
8. Check convergence: $\|\mathbf{A}\mathbf{x}^{(k)\star}\|_1 < \tau_{rel}$, otherwise set $k \leftarrow k+1$ and go to 2
```

Remark 4.2.1. Depending on the functional F and the optimization routine it may not be necessary to explicitly compute $\mathbf{X}\mathbf{z}^\star$ and $\|\mathbf{A}\mathbf{x}^{(k)\star}\|_1$ in Algorithm 4.3, as these quantities are often available from the minimization process. In the case of two-norm minimization it may be sufficient to reject the improved iterate using the stricter condition

$$\|\mathbf{A}\mathbf{x}^{(k)\star}\|_2 \geq \|\mathbf{r}^{(k)}\|_1,$$

to help reduce the computational overhead.

4.3 Two-norm minimization

As discussed in the previous section, when $F(\mathbf{u}) = \|\mathbf{A}\mathbf{u}\|_2$ the iterant recombination minimization problem can be cast as a quadratic program. To solve this problem we use Matlab's built-in quadratic programming solver `quadprog`. In particular, we use the medium-scale version of this algorithm which is based on an *active-set method* (see [94] for details). Active-set methods attempt to identify which inequality constraints are *active* at the solution (that is, which inequality constraints are equal to zero at the solution), and treat the active constraints as equalities in the subproblems that define the iterates [59]. In general, active-set methods find a step from one iterate to the next by solving a quadratic subproblem in which all the equality constraints and some of the inequality constraints are imposed as equalities. This subset is referred to as the *working set*, which is updated by adding and removing constraints as the algorithm proceeds. Thus, active-set methods are attractive for the minimization of (4.13) because only a few of the n inequality constraints may be relevant.

When the window size m equals two we can directly compute the analytic solution of (4.13). For simplicity we work with the equivalent inequality-form problem (see (4.15))

$$\begin{array}{ll} \text{minimize} & g(z) = \|(\mathbf{r}_2 - \mathbf{r}_1)z + \mathbf{r}_1\|_2 \\ \text{subject to} & (\mathbf{x}_1 - \mathbf{x}_2)z \leq \mathbf{x}_1 - \delta x_{min}\mathbf{1}, \end{array} \tag{4.16}$$

where $\mathbf{r}_1 = \mathbf{A}\mathbf{x}_1$, $\mathbf{r}_2 = \mathbf{A}\mathbf{x}_2$, and $z \in \mathbb{R}$. If $z^\star$ is the optimal solution of (4.16), then $\mathbf{z}^\star = (1 - z^\star, z^\star)^\top$ is the optimal solution of the corresponding two-dimensional problem. Defining the index sets

$$\mathcal{I}^+ = \{i : x_{i1} - x_{i2} > 0\} \quad \text{and} \quad \mathcal{I}^- = \{i : x_{i1} - x_{i2} < 0\},$$

the constraint $(\mathbf{x}_1 - \mathbf{x}_2)z \leq \mathbf{x}_1 - \delta x_{min}\mathbf{1}$ implies that

$$-\infty \leq L = \sup_{i \in \mathcal{I}^-} \left(\frac{x_{i1} - \delta x_{min}}{x_{i1} - x_{i2}} \right) \leq z^\star \leq \inf_{i \in \mathcal{I}^+} \left(\frac{x_{i1} - \delta x_{min}}{x_{i1} - x_{i2}} \right) = U \leq +\infty.$$

We note that $L < 0 < U$ always holds, so the feasible set is never empty. Assuming that $\mathbf{r}_1 \neq \mathbf{r}_2$, the function g is a concave up parabola, and hence the minimizer occurs at the vertex of the parabola. Thus, the strict global minimizer of g over $\mathbb{R}$ is given by

$$z_{gbl} = \frac{\langle \mathbf{r}_1, \mathbf{r}_1 \rangle - \langle \mathbf{r}_1, \mathbf{r}_2 \rangle}{\langle \mathbf{r}_1 - \mathbf{r}_2, \mathbf{r}_1 - \mathbf{r}_2 \rangle}.$$

If $\mathbf{r}_1 = \mathbf{r}_2$, then $\mathbf{x}_1 = \mathbf{x}_2$ in which case we select $z^\star = 1$ as the minimizer. When $\mathbf{r}_1 \neq \mathbf{r}_2$, we choose

$$z^\star = \begin{cases} z_{gbl} & \text{if } z_{gbl} \in [L, U], \\ L & \text{if } |z_{gbl} - L| < |z_{gbl} - U|, \\ U & \text{otherwise.} \end{cases}$$

Since the computed values of L and U, denoted by $\tilde{L}$ and $\tilde{U}$, are susceptible to roundoff error, we replace them by $\tilde{L}(1+\omega)$ and $\tilde{U}(1-\omega)$, respectively, for some sufficiently small $\omega > 0$. The purpose of using ω is to prevent the computed value of $z^\star$ from falling outside the feasible set. In practice $\omega = 10^{-14}$ seems to be a suitable value. If the matrix $\mathbf{AX}$ has already been updated, then computing z_{gbl} requires $9n$ flops to leading order, assuming that the searches to find L and U require $2n$ flops in total.

4.4 The ellipsoid method

The ellipsoid method was first described in 1976 by Iudin and Nemirovskii [71], and was explicitly stated as we know it today in 1977 by Shor [106]. It gained notoriety in the early 1980s when Khachiyan showed that a variation of the ellipsoid method for linear optimization could be implemented with polynomial time complexity [74]. Although the ellipsoid method was not competitive in practice for linear optimization, it has shown itself to be a robust solver for nonlinear convex programs, which at some levels of solution error is competitive with other more mainstream solvers [57].

The ellipsoid method was originally intended as a solver for nonlinear convex optimization problems of the form

$$\begin{aligned} \text{minimize} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{S} = \{\mathbf{y} \in \mathbb{R}^m : f_i(\mathbf{y}) \leq 0,\ i = 1, \ldots, n\}, \end{aligned} \tag{4.17}$$

where each $f_i : \mathbb{R}^m \to \mathbb{R}$ for $i = 0, \ldots, n$ is a finite convex function on $\mathbb{R}^m$ that is not required to be differentiable. Here, the function f_0 is referred to as the *objective function* and the functions $f_1, \ldots, f_n$ are referred to as *constraint functions*. The set $\mathcal{S}$ of all points that satisfy the n inequality constraints is called the *feasible set*. Hence, any point belonging to $\mathcal{S}$ is called *feasible*, and any point not belonging to $\mathcal{S}$ is called *infeasible*. We assume that the feasible set is nonempty and that there exists an optimal solution $\mathbf{x}^\star$ to (4.17).

Definition 4.4.1 (Ellipsoid)**.** Let $\mathbf{D}$ be an $m \times m$ symmetric positive definite matrix and let $\mathbf{x}_0$ be any point in $\mathbb{R}^m$. Then the set

$$\mathcal{E}(\mathbf{x}_0, \mathbf{D}) = \{\mathbf{x} \in \mathbb{R}^n : (\mathbf{x} - \mathbf{x}_0)^\top \mathbf{D}^{-1}(\mathbf{x} - \mathbf{x}_0) \leq 1\}$$

is an ellipsoid with center $\mathbf{x}_0$.

Definition 4.4.2 (Hyperplane and Halfspace)**.** Let $\mathbf{a}$ be a nonzero vector in $\mathbb{R}^m$ and c a scalar.

1. The set $\{\mathbf{x} \in \mathbb{R}^m : \mathbf{a}^\top \mathbf{x} = c\}$ is a hyperplane.

2. The set $\{\mathbf{x} \in \mathbb{R}^m : \mathbf{a}^\top \mathbf{x} \leq c\}$ is a halfspace.

Suppose we have an initial ellipsoid $\mathcal{E}^{(0)}$ that contains $\mathbf{x}^\star$. The ellipsoid method iteratively constructs a sequence of successively "smaller" ellipsoids each of which contains $\mathbf{x}^\star$. By smaller, we mean that the volume of the next ellipsoid is strictly less than the volume of the previous ellipsoid. Suppose now that $\mathcal{E}^{(k)} = \mathcal{E}(\mathbf{x}^{(k)}, \mathbf{D}^{(k)})$ is the kth ellipsoid in this sequence. Then $\mathcal{E}^{(k+1)}$ may be constructed as follows. Determine a hyperplane that passes through $\mathbf{x}^{(k)}$. Then $\mathbf{x}^\star$ is contained in one of the halfspaces generated by this hyperplane, call it $\mathcal{H}^{(k)}$. Now define $\mathcal{E}^{(k+1)} = \mathcal{E}(\mathbf{x}^{(k+1)}, \mathbf{D}^{(k+1)})$ as the *minimum volume ellipsoid* that contains the intersection of $\mathcal{E}^{(k)}$ with $\mathcal{H}^{(k)}$. Since $\mathbf{x}^\star \in (\mathcal{E}^{(k)} \cap \mathcal{H}^{(k)})$, it follows that $\mathbf{x}^\star \in \mathcal{E}^{(k+1)}$. Furthermore, if the center point $\mathbf{x}^{(k+1)}$ is feasible, then it is an approximation of $\mathbf{x}^\star$. This procedure is illustrated in Figure 4.2. If the hyperplane passes through $\mathbf{x}^{(k)}$, then $\mathcal{E}^{(k+1)}$ is a *center-cut* ellipsoid. If instead the hyperplane passes between $\mathbf{x}^{(k)}$ and $\mathbf{x}^\star$, then $\mathcal{E}^{(k+1)}$ is a *deep-cut* ellipsoid [55]. Intuitively, it is clear that the deep-cut ellipsoid contains $\mathbf{x}^\star$ but less than half of $\mathcal{E}^{(k)}$.

The normal vector $\mathbf{g}^{(k)}$ that defines the hyperplane through $\mathbf{x}^{(k)}$, that is, $\{\mathbf{y} \in \mathbb{R}^m : {\mathbf{g}^{(k)}}^\top(\mathbf{y} - \mathbf{x}^{(k)}) = 0\}$, is chosen in such a way that it is easy to decide in which halfspace

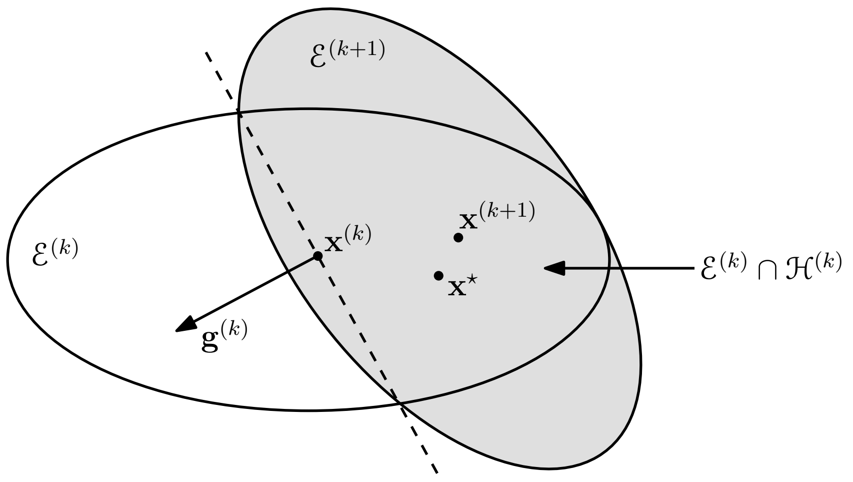


Figure 4.2: Construction of the minimum volume ellipsoid $\mathcal{E}^{(k+1)}$. Here $\mathbf{g}^{(k)}$ is the normal vector to the hyperplane (dashed line).

$\mathbf{x}^\star$ is located. First, we require the definition of a *subgradient* and *subdifferential*, which generalize the derivative to functions that are not differentiable.

Definition 4.4.3 (Subgradient, Subdifferential)**.** Let $f : \mathcal{C} \to \mathbb{R}$ be a convex function whose domain is an open convex set $\mathcal{C} \subset \mathbb{R}^m$, and let $\mathbf{x}_0 \in \mathcal{C}$. Then the vector $\mathbf{g}$ is a subgradient of f at $\mathbf{x}_0$ if

$$f(\mathbf{x}_0) + \mathbf{g}^\top(\mathbf{x} - \mathbf{x}_0) \leq f(\mathbf{x}) \quad \text{for all} \quad \mathbf{x} \in \mathcal{C}.$$

The set of all subgradients of f at $\mathbf{x}_0$ is denoted by $\partial f(\mathbf{x}_0)$ and is called the subdifferential of f at $\mathbf{x}_0$. If f is convex and differentiable at $\mathbf{x}_0$ then $\partial f(\mathbf{x}_0) = \{\nabla f(\mathbf{x}_0)\}$.

There are two possibilities to consider for the center-cut algorithm. If the current approximation $\mathbf{x}^{(k)}$ is infeasible, that is, if there exists an index $j > 0$ such that $f_j(\mathbf{x}^{(k)}) > 0$, then we choose $\mathbf{g}^{(k)} \in \partial f_j(\mathbf{x}^{(k)})$. Otherwise, if $\mathbf{x}^{(k)}$ is feasible, then we choose $\mathbf{g}^{(k)} \in \partial f_0(\mathbf{x}^{(k)})$. In either case it is straightforward to verify that

$$\mathbf{x}^\star \in \mathcal{H}^{(k)} = \{\mathbf{y} \in \mathbb{R}^m : {\mathbf{g}^{(k)}}^\top(\mathbf{y} - \mathbf{x}^{(k)}) \leq 0\}.$$

In order to define a hyperplane it is necessary to find a nonzero subgradient vector. To show that a nonzero subgradient exists, we start with the fact that since each f_i is convex on $\mathbb{R}^m$, it has a nonempty subdifferential at any point in $\mathbb{R}^m$ [7]. Now it is still possible that the subdifferential of f_i at some point in $\mathbb{R}^m$ contains only the zero vector. By the definition of the subgradient, for any iterate $\mathbf{x}^{(k)}$ we have that

$$\mathbf{g}^{(k)\top}(\mathbf{x} - \mathbf{x}^{(k)}) \leq f_i(\mathbf{x}) - f_i(\mathbf{x}^{(k)}) \quad \text{for all} \quad \mathbf{x} \in \mathbb{R}^m \quad \text{and} \quad \mathbf{g}^{(k)} \in \partial f_i(\mathbf{x}^{(k)}). \tag{4.18}$$

In the infeasible case, $f_i(\mathbf{x}^{(k)}) > 0$ for some $i \in \{1, \ldots, n\}$. If $\partial f_i(\mathbf{x}^{(k)}) = \{\mathbf{0}\}$, then by (4.18)

$$0 < f_i(\mathbf{x}^{(k)}) \leq f_i(\mathbf{x}) \quad \text{for all} \quad \mathbf{x} \in \mathbb{R}^m. \tag{4.19}$$

However, this inequality implies the feasible set $\mathcal{S}$ is empty, which contradicts our assumption that $\mathcal{S}$ is nonempty. In the feasible case, if $\partial f_0(\mathbf{x}^{(k)}) = \{\mathbf{0}\}$, then it follows from (4.18) that $\mathbf{x}^{(k)}$ is optimal. Therefore, if $\mathbf{x}^{(k)}$ is feasible but not optimal a nonzero subgradient must exist.

It remains to describe the update equations for $\mathcal{E}^{(k+1)}$. Given $\mathcal{E}^{(k)} = \mathcal{E}(\mathbf{x}^{(k)}, \mathbf{D}^{(k)}) \in \mathbb{R}^m$ with $m > 1$ and the subgradient vector $\mathbf{g}^{(k)}$ defining the halfspace $\mathcal{H}^{(k)}$, the center-cut minimum volume ellipsoid that contains the region $\mathcal{E}^{(k)} \cap \mathcal{H}^{(k)}$ is given by

$$\mathcal{E}^{(k+1)} = \mathcal{E}(\mathbf{x}^{(k+1)}, \mathbf{D}^{(k+1)}),$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \frac{1}{m+1} \frac{\mathbf{D}^{(k)}\mathbf{g}^{(k)}}{\sqrt{\mathbf{g}^{(k)\top}\mathbf{D}^{(k)}\mathbf{g}^{(k)}}}, \tag{4.20a}$$

$$\mathbf{D}^{(k+1)} = \frac{m^2}{m^2 - 1}\left(\mathbf{D}^{(k)} - \frac{2}{m+1} \frac{\mathbf{D}^{(k)}\mathbf{g}^{(k)}(\mathbf{D}^{(k)}\mathbf{g}^{(k)})^\top}{\mathbf{g}^{(k)\top}\mathbf{D}^{(k)}\mathbf{g}^{(k)}}\right). \tag{4.20b}$$

Indeed, it can be verified that $\mathbf{D}^{(k+1)}$ is symmetric positive definite, and that the volume of $\mathcal{E}^{(k+1)}$ is strictly less than the volume of $\mathcal{E}^{(k)}$. A rigorous derivation of these results is given in [12] (see also [14]). In the one-dimensional case ($m = 1$) we have

$$x^{(k+1)} = x^{(k)} - \frac{1}{2}\operatorname{sgn}(g^{(k)})\sqrt{D^{(k)}}, \tag{4.21a}$$

$$D^{(k+1)} = \frac{D^{(k)}}{4}, \tag{4.21b}$$

where $\operatorname{sgn}(\cdot)$ is the signum function and all quantities are scalar. In this case the ellipsoid method is equivalent to the bisection method.

The update formulas for an ellipsoid method with deep cuts are very similar to those given in (4.20). Since we make use of deep cuts in our numerical tests, we briefly describe them below. Let $\mathcal{E} = \mathcal{E}(\mathbf{x}_0, \mathbf{D})$ be an ellipsoid in $\mathbb{R}^m$. It was shown in [57] that any hyperplane given by $\{\mathbf{y} \in \mathbb{R}^m : \mathbf{g}^\top \mathbf{y} = \beta\}$ with $\beta = \mathbf{g}^\top \mathbf{x}_0 - \alpha\sqrt{\mathbf{g}^\top \mathbf{D}\mathbf{g}}$ and $\alpha \in [-1, 1]$ has a nonempty intersection with $\mathcal{E}$. Furthermore, for $\alpha \in [-1/m, 1]$ it is possible to construct a minimum volume ellipsoid that contains the intersection of $\mathcal{E}$ and the halfspace

$$\mathcal{H} = \left\{\mathbf{y} \in \mathbb{R}^m : \mathbf{g}^\top(\mathbf{y} - \mathbf{x}_0) \leq -\alpha\sqrt{\mathbf{g}^\top \mathbf{D}\mathbf{g}}\right\}. \tag{4.22}$$

We note that if $\alpha < -1$ then $\mathcal{E} \cap \mathcal{H} = \mathcal{H}$, if $\alpha > 1$ then $\mathcal{E} \cap \mathcal{H} = \emptyset$, and if $-1 \leq \alpha < -1/m$ then $\mathcal{E}$ is the smallest ellipsoid containing $\mathcal{E} \cap \mathcal{H}$ [14]. For $m > 1$ define the parameters:

$$\tau := \frac{1 + \alpha m}{m + 1}, \quad \sigma := \frac{2(1 + \alpha m)}{(m + 1)(1 + \alpha)}, \quad \delta := \frac{m^2(1 - \alpha^2)}{m^2 - 1}. \tag{4.23}$$

Then according to [14] the deep-cut ellipsoid with volume strictly less than $\mathcal{E}^{(k)}$ is given by

$$\mathcal{E}^{(k+1)} = \mathcal{E}(\mathbf{x}^{(k+1)}, \mathbf{D}^{(k+1)}),$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \tau \frac{\mathbf{D}^{(k)}\mathbf{g}^{(k)}}{\sqrt{\mathbf{g}^{(k)\top}\mathbf{D}^{(k)}\mathbf{g}^{(k)}}}, \tag{4.24a}$$

$$\mathbf{D}^{(k+1)} = \delta\left(\mathbf{D}^{(k)} - \sigma \frac{\mathbf{D}^{(k)}\mathbf{g}^{(k)}(\mathbf{D}^{(k)}\mathbf{g}^{(k)})^\top}{\mathbf{g}^{(k)\top}\mathbf{D}^{(k)}\,\mathbf{g}^{(k)}}\right). \tag{4.24b}$$

In the one-dimensional case

$$x^{(k+1)} = x^{(k)} - \frac{(1+\alpha)}{2}\operatorname{sgn}(g^{(k)})\sqrt{D^{(k)}}, \tag{4.25a}$$

$$D^{(k+1)} = \frac{(1-\alpha)^2}{4}D^{(k)}. \tag{4.25b}$$

The parameter α in the equations above determines the depth of the cut. If $\alpha \in [-1/m, 0)$ then $\mathcal{E}^{(k+1)}$ is referred to as a *shallow-cut* ellipsoid, and $\mathcal{E}^{(k+1)}$ contains more than half of $\mathcal{E}^{(k)} \cap \mathcal{H}^{(k)}$ including $\mathbf{x}^{(k)}$ [14]. For $\alpha = 0$ we recover the formulas for the center-cut ellipsoid in (4.20) and (4.21), and for $\alpha \in (0, 1]$ we obtain a deep-cut ellipsoid. In our implementation α_k (the depth of cut on the kth iteration) is computed as follows. If $\mathbf{x}^{(k)}$ is feasible then

$$\alpha_k = (f_0(\mathbf{x}^{(k)}) - u_k)\left(\mathbf{g}^{(k)\top}\mathbf{D}^{(k)}\mathbf{g}^{(k)}\right)^{-1/2} \tag{4.26}$$

with

$$u_k = \min\{f_0(\mathbf{x}^{(i)}) : 1 \le i \le k,\, \mathbf{x}^{(i)} \in \mathcal{S}\}. \tag{4.27}$$

Otherwise, if $f_j(\mathbf{x}^{(k)}) > 0$ for some index $j > 0$, then

$$\alpha_k = f_j(\mathbf{x}^{(k)})\left(\mathbf{g}^{(k)\top}\mathbf{D}^{(k)}\mathbf{g}^{(k)}\right)^{-1/2}. \tag{4.28}$$

It was shown in [57] that computing α_k according to formulas (4.26) and (4.28) always yields a valid cut. For further details regarding deep cuts as well as examples of their use in deep-cut ellipsoid methods we refer to [54, 57].

A stopping criterion for the ellipsoid method is given by

$$u_k - l_k < \varepsilon, \tag{4.29}$$

where l_k and u_k are the current best upper and lower bounds for the optimal objective value, that is, $l_k \le f_0(\mathbf{x}^\star) \le u_k$. The upper bound u_k is given by (4.27), and

$$l_k = \max\left\{ f_0(\mathbf{x}^{(i)}) - \sqrt{\mathbf{g}^{(i)\top}\mathbf{D}^{(i)}\,\mathbf{g}^{(i)}} : \mathbf{x}^{(i)} \in \mathcal{S},\, 1 \le i \le k \right\}.$$

The lower bound l_k can be derived as follows. For any feasible iterate $\mathbf{x}^{(k)}$, it follows by the subgradient inequality that

$$\begin{aligned} f_0(\mathbf{x}^\star) &\ge f_0(\mathbf{x}^{(k)}) + \mathbf{g}^{(k)\top}(\mathbf{x}^\star - \mathbf{x}^{(k)}) \\ &\ge f_0(\mathbf{x}^{(k)}) + \inf_{\mathbf{z}\in\mathcal{E}^{(k)}} \mathbf{g}^{(k)\top}(\mathbf{z} - \mathbf{x}^{(k)}). \end{aligned} \tag{4.30}$$

Since $\mathcal{E}^{(k)}$ is a compact subset of $\mathbb{R}^m$ and $\mathbf{g}^{(k)\top}(\mathbf{z} - \mathbf{x}^{(k)})$ is continuous, the infimum is attained on the boundary of $\mathcal{E}^{(k)}$. The minimizer of (4.30) can then be obtained through a straightforward application of Lagrange multipliers. At convergence

$$0 \le u_k - f_0(\mathbf{x}^\star) \le u_k - l_k < \varepsilon,$$

and the feasible iterate $\mathbf{x}_{best}$ that satisfies $f_0(\mathbf{x}_{best}) = u_k$ is returned. We note that since the ellipsoid method is not a descent method it is necessary to keep track of the best feasible iterate discovered throughout the course of the algorithm.

By arguments similar to those for deriving the lower bound l_k, if $\mathbf{x}^{(k)}$ is infeasible, then for some index $j > 0$

$$f_j(\mathbf{x}) \ge f_j(\mathbf{x}^{(k)}) - \sqrt{\mathbf{g}^{(k)\top}\mathbf{D}^{(k)}\,\mathbf{g}^{(k)}} \quad \text{for all} \quad \mathbf{x} \in \mathcal{E}^{(k)}.$$

Therefore, if the right-hand side of this expression is strictly positive, the convex optimization problem (4.17) is infeasible. In this case no further progress can be made, and execution of the algorithm is terminated.

A pseudocode description of the deep-cut ellipsoid method is given by Algorithm 4.4. Due to numerical roundoff in finite precision arithmetic, the computed matrix $\mathbf{D}^{(k)}$ will invariably become indefinite. Consequently, the quantity γ may not be a real number.

Fortunately, numerical stability of Algorithm 4.4 can be remedied in the following way. As advocated in [14], the matrix $\mathbf{D}^{(0)}$ can be factorized into its Cholesky factors $\mathbf{D}^{(0)} = \mathbf{L}^{(0)}\mathbf{\Lambda}^{(0)}\mathbf{L}^{(0)\top}$, where $\mathbf{L}^{(0)}$ is lower triangular and $\mathbf{\Lambda}^{(0)}$ is a positive definite diagonal matrix. (Algorithm 4.1.2 in [60] is a numerically stable implementation for computing the Cholesky factors of a symmetric positive definite matrix.) Then, $\mathbf{D}^{(k)}$ can be maintained in product form by updating $\mathbf{L}^{(k)}$ and $\mathbf{\Lambda}^{(k)}$. Referring to Algorithm 4.4, the matrix $\mathbf{D}^{(k+1)}$ is obtained through a symmetric rank-one modification of $\mathbf{D}^{(k)}$. In [58], Gill, Golub, Murray, and Saunders discuss algorithms for computing the Cholesky factors of a symmetric positive definite matrix modified by a symmetric matrix of rank one. Of the methods discussed in [58], we use algorithm C2 to compute $\mathbf{L}^{(k+1)}$ and $\mathbf{\Lambda}^{(k+1)}$, primarily because of its good numerical stability. The update process for $\mathbf{D}^{(k)}$ requires approximately $3m^2 + \mathcal{O}(m)$ flops and $m + 1$ square roots. The only other modification of Algorithm 4.4 is the computation of the vector $\mathbf{g}$ on line 9, which is given by the following sequence of steps:

$$\mathbf{u} = \mathbf{L}^{(k)\top}\mathbf{g}^{(k)}, \quad \mathbf{v} = \mathbf{\Lambda}^{(k)}\mathbf{u}, \quad \gamma = \sqrt{\mathbf{u}^\top\mathbf{v}}, \quad \mathbf{g} = \gamma^{-1}\mathbf{L}^{(k)}\mathbf{v}. \tag{4.31}$$

We note that the formula for γ is equivalent to

$$\gamma = \sqrt{\mathbf{u}^\top\mathbf{v}} = \sqrt{\mathbf{u}^\top\mathbf{\Lambda}^{(k)}\mathbf{u}}\,,$$

which must be a real number because $\mathbf{\Lambda}^{(k)}$ is a positive definite diagonal matrix.

We conclude this section with a brief discussion regarding convergence of the ellipsoid algorithm. Convergence of the center-cut variant of Algorithm 4.4 ($\alpha = 0$) applied to convex programming problems was proved in [63] using an approach based on variational inequalities. Moreover, Frenk, Gromicho and Zhang [57] proved convergence of center-cut and deep-cut variants of the ellipsoid method applied to the convex program (4.17). Here, convergence is understood in the following sense:

$$\lim_{k\to\infty} u_k = f_0(\mathbf{x}^\star).$$

The speed of convergence of the center-cut ellipsoid method is geometric with rate

$$\left(\frac{m^2}{m^2-1}\sqrt[m]{\frac{m-1}{m+1}}\right)^{1/2} \quad \text{for} \quad m \geq 2, \tag{4.32}$$

which approaches unity as $m \to \infty$. The convergence rate attains its minimum value of $1/2$ when $m = 1$, in which case the ellipsoid method is equivalent to the bisection method.

Algorithm 4.4: Deep-cut ellipsoid method

```
 1. Let $\mathcal{E}^{(0)} = \mathcal{E}(\mathbf{x}^{(0)}, \mathbf{D}^{(0)})$ be an initial ellipsoid such that $\mathbf{x}^\star \in \mathcal{E}^{(0)}$
 2. Set $k \leftarrow 0$
    while $k < K$ do
        if $f_j(\mathbf{x}^{(k)}) > 0$ for some index $j > 0$ then
 3.         Choose $\mathbf{g}^{(k)} \in \partial f_j(\mathbf{x}^{(k)})$
 4.         Set $\gamma \leftarrow \left(\mathbf{g}^{(k)\top}\mathbf{D}^{(k)}\mathbf{g}^{(k)}\right)^{1/2}$ and compute $\alpha$ according to (4.26)
        else
 5.         Choose $\mathbf{g}^{(k)} \in \partial f_0(\mathbf{x}^{(k)})$
 6.         Set $\gamma \leftarrow \left(\mathbf{g}^{(k)\top}\mathbf{D}^{(k)}\mathbf{g}^{(k)}\right)^{1/2}$ and compute $\alpha$ according to (4.28)
        end
 7.     Set $\mathbf{g} \leftarrow \gamma^{-1}\mathbf{D}^{(k)}\mathbf{g}^{(k)}$
 8.     If $\mathbf{x}^{(k)}$ is feasible, check the stopping criterion, otherwise check for infeasibility
 9.     Compute $\tau$, $\delta$, and $\sigma$ according to (4.23)
10.     Construct a new ellipsoid $\mathcal{E}^{(k+1)} = \mathcal{E}(\mathbf{x}^{(k+1)}, \mathbf{D}^{(k+1)})$ with

            $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \tau\mathbf{g}$ and $\mathbf{D}^{(k+1)} \leftarrow \delta\left(\mathbf{D}^{(k)} - \sigma\mathbf{g}\mathbf{g}^\top\right)$

11.     Set $k \leftarrow k + 1$
    end
```

4.5 One-norm minimization via the ellipsoid method

In this section we discuss how the ellipsoid method can be applied as a solver for the iterant recombination problem. In particular, formulas for the subgradients of the objective function and the constraint functions are given, and it is shown how an initial ellipsoid $\mathcal{E}^{(0)}$ that contains the exact solution of the minimization problem can be constructed.

We work with the inequality-form problem given by

$$\begin{aligned} \text{minimize} \quad & \|\hat{\mathbf{A}}\hat{\mathbf{z}} + \mathbf{r}_1\| \\ \text{subject to} \quad & \hat{\mathbf{X}}\hat{\mathbf{z}} - \mathbf{x}_1 \leq \mathbf{0} \\ & \hat{\mathbf{z}} \in \mathbb{R}^{m-1}, \end{aligned} \tag{4.33}$$

where $\|\cdot\|$ is a norm on $\mathbb{R}^n$. We note that since the ellipsoid method is an *interior-point method*, its feasible iterates approach $\mathbf{x}^\star$ from the interior of the feasible region. Therefore, the solution of (4.33) by the ellipsoid method has strictly positive components. It is clear that the objective function of (4.33) is

$$f_0(\hat{\mathbf{z}}) = \|\hat{\mathbf{A}}\hat{\mathbf{z}} + \mathbf{r}_1\|$$

and the constraint functions are

$$f_i(\hat{\mathbf{z}}) = \hat{\mathbf{x}}_i^\top \hat{\mathbf{z}} - (\mathbf{x}_1)_i \quad \text{for} \quad i = 1, \ldots, n,$$

where $\hat{\mathbf{x}}_i$ is the ith column of $\hat{\mathbf{X}}^\top$. Since the constraint functions are convex and differentiable with respect to $\hat{\mathbf{z}}$, the subdifferential $\partial f_i(\hat{\mathbf{z}}) = \{\nabla f_i(\hat{\mathbf{z}})\} = \{\hat{\mathbf{x}}_i\}$ for all $\hat{\mathbf{z}} \in \mathbb{R}^{m-1}$. Treating the objective function as a composition of $\hat{\mathbf{A}}\hat{\mathbf{z}} + \mathbf{r}_1$ and $h(\cdot) = \|\cdot\|$, it follows by the (subgradient) chain rule [102] that

$$\partial f_0(\hat{\mathbf{z}}) = \hat{\mathbf{A}}^\top \partial h(\hat{\mathbf{A}}\hat{\mathbf{z}} + \mathbf{r}_1).$$

We note that in the case of one-norm minimization the vector function $\mathbf{q} \in \mathbb{R}^n$ defined by

$$q_i(\mathbf{x}) = \begin{cases} 1 & \text{if } x_i \geq 0, \\ -1 & \text{if } x_i < 0 \end{cases} \tag{4.34}$$

is a subgradient for $h(\mathbf{x}) = \|\mathbf{x}\|_1$. Furthermore, if $h(\mathbf{x}) = \|\mathbf{x}\|_2$ (quadratic minimization) the subgradient of h is given by $\nabla h(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|_2$.

We now describe a procedure to find an initial ellipsoid $\mathcal{E}^{(0)} = \mathcal{E}(\hat{\mathbf{z}}^{(0)}, \mathbf{D}^{(0)})$ that is guaranteed to contain $\hat{\mathbf{z}}^\star$, assuming that $\hat{\mathbf{z}}^\star$ exists. In addition, we state necessary and sufficient conditions for the existence of $\mathcal{E}^{(0)}$. Intuitively, we expect that most of the weight in the optimal linear combination $\mathbf{x}^\star = \mathbf{X}\mathbf{z}^\star$ will be associated with the most recent fine-level approximation $\mathbf{x}^{(k)}$, which is the rightmost column of $\mathbf{X}$. Therefore, we choose

$$\hat{\mathbf{z}}^{(0)} = (0, \ldots, 0, 1)^\top$$

as the center point for the initial ellipsoid. We now derive the matrix $\mathbf{D}^{(0)}$. If $\mathbf{z}$ is any feasible point of (4.12) then the corresponding point $\hat{\mathbf{z}}$ is feasible for (4.33), and $\mathbf{A}\mathbf{X}\mathbf{z} = \hat{\mathbf{A}}\hat{\mathbf{z}} + \mathbf{r}_1$. Therefore, given some feasible point $\mathbf{z} \in \mathbb{R}^m$ it follows that

$$\hat{\mathbf{z}}^\star \in \{\mathbf{y} \in \mathbb{R}^{m-1} : \|\hat{\mathbf{A}}\mathbf{y} + \mathbf{r}_1\| \leq \alpha\} \quad \text{with} \quad \alpha = \|\mathbf{A}\mathbf{X}\mathbf{z}\| = \|\hat{\mathbf{A}}\hat{\mathbf{z}} + \mathbf{r}_1\|.$$

For example, the ith canonical basis vector in $\mathbb{R}^m$, denoted by $\mathbf{e}_i$, is a feasible point for (4.12). In practice we choose α according to

$$\alpha = \min_{i=1,\ldots,m} \|\mathbf{A}\mathbf{X}\mathbf{e}_i\|.$$

Since $\hat{\mathbf{A}}\mathbf{y} + \mathbf{r}_1 = \hat{\mathbf{A}}(\mathbf{y} - \hat{\mathbf{z}}^{(0)}) + (\hat{\mathbf{A}}\hat{\mathbf{z}}^{(0)} + \mathbf{r}_1)$ for any $\mathbf{y} \in \mathbb{R}^{m-1}$, it is clear that

$$\hat{\mathbf{z}}^\star \in \{\mathbf{y} \in \mathbb{R}^{m-1} : \|\hat{\mathbf{A}}(\mathbf{y} - \hat{\mathbf{z}}^{(0)}) + (\hat{\mathbf{A}}\hat{\mathbf{z}}^{(0)} + \mathbf{r}_1)\| \leq \alpha\}.$$

Applying the reverse triangle inequality, $\big|\|\mathbf{u}\| - \|\mathbf{v}\|\big| \leq \|\mathbf{u} \pm \mathbf{v}\|$, we obtain

$$\hat{\mathbf{z}}^\star \in \{\mathbf{y} \in \mathbb{R}^{m-1} : \|\hat{\mathbf{A}}(\mathbf{y} - \hat{\mathbf{z}}^{(0)})\| \leq r\} \quad \text{with} \quad r = \alpha + \|\hat{\mathbf{A}}\hat{\mathbf{z}}^{(0)} + \mathbf{r}_1\|.$$

Since $(1/\beta)\|\cdot\|_2 \leq \|\cdot\|$ for some $\beta > 0$ (equivalence of norms on $\mathbb{R}^n$) we arrive at the desired result:

$$\|\hat{\mathbf{A}}(\hat{\mathbf{z}}^\star - \hat{\mathbf{z}}^{(0)})\|_2 \leq \beta r \quad \Leftrightarrow \quad (\hat{\mathbf{z}}^\star - \hat{\mathbf{z}}^{(0)})^\top \mathbf{D}^{(0)^{-1}} (\hat{\mathbf{z}}^\star - \hat{\mathbf{z}}^{(0)}) \leq 1, \tag{4.35}$$

where $\mathbf{D}^{(0)} = \beta^2 r^2 (\hat{\mathbf{A}}^\top \hat{\mathbf{A}})^{-1}$. Therefore, the optimal solution $\hat{\mathbf{z}}^\star$ belongs to the ellipsoid $\mathcal{E}^{(0)} = \mathcal{E}(\hat{\mathbf{z}}^{(0)}, \mathbf{D}^{(0)})$. We note that $\beta = 1$ for $\|\cdot\| = \|\cdot\|_1$.

In order for the initial ellipsoid described above to exist, the matrix $\hat{\mathbf{A}}^\top\hat{\mathbf{A}}$ must be invertible. It is clear that $\hat{\mathbf{A}}^\top\hat{\mathbf{A}}$ is an $(m-1)\times(m-1)$ symmetric positive semidefinite matrix, and provided that $\hat{\mathbf{A}}$ is of full rank, $\mathbf{D}^{(0)}$ exists and is symmetric positive definite. Therefore, we must determine under what conditions $\hat{\mathbf{A}}$ is of full rank. We begin with a simple observation.

Proposition 4.5.1. *The exact solution of* $\mathbf{A}\mathbf{x} = \mathbf{0}$ *does not belong to the range of* $\hat{\mathbf{X}}$.

Proof. Suppose to the contrary that $\mathbf{x} \in \text{range}(\hat{\mathbf{X}})$. Then there exists some vector $\mathbf{y}$ such that

$$\mathbf{x} = \hat{\mathbf{X}}\mathbf{y}.$$

However, this result implies that

$$\mathbf{1}^\top \mathbf{x} = \mathbf{1}^\top \hat{\mathbf{X}}\mathbf{y} = 0,$$

which is not possible because $\mathbf{x}$ is a probability vector. □

The above proposition establishes that $\hat{\mathbf{A}}$ has full rank if and only if $\hat{\mathbf{X}}$ has full rank since $\hat{\mathbf{A}} = -\mathbf{A}\hat{\mathbf{X}}$. To handle the situation in which $\hat{\mathbf{X}}$ is rank deficient we employ the following simple strategy. If $\hat{\mathbf{X}}$ is rank deficient then $\mathbf{X}$ is also rank deficient. Thus, the most obvious approach is to drop all columns of $\mathbf{X}$ except for the rightmost column corresponding to the most recent multigrid iterate, and to skip the next iterant recombination step. Doing so it follows that $\mathbf{X}$ has full rank because it consists of a single nonzero column. The iterant recombination process can then be restarted. In practice, rank deficiency of $\hat{\mathbf{A}}$ is rarely an issue because of the nonlinear nature of the underlying multilevel algorithm, that is, because the range of the multilevel iteration operator changes with each iteration.

We conclude this section by discussing the computational costs of using the ellipsoid method in conjunction with the iterant recombination process. The main per iteration computational costs of Algorithm 4.4 are the subgradient vector construction (line 3), and the ellipsoid update (line 5). Construction of the subgradient vector consists of four steps: (1) Perform a feasibility check, (2) compute $\hat{\mathbf{A}}\hat{\mathbf{z}} + \mathbf{r}_1$, (3) compute $\mathbf{q}$, (4) compute $\hat{\mathbf{A}}^\top \mathbf{q}$. The feasibility check consists of evaluating $\hat{\mathbf{X}}\hat{\mathbf{z}} - \mathbf{x}_1$ and then searching for a positive entry, which requires $\mathcal{O}(mn)$ flops. The order in which the feasibility constraints are examined is discussed in [54], where the authors advocate a cyclical order because it yields slightly better efficiency. However, in our implementation we use a straightforward top-down sequential search of $f_1, \dots, f_n$, which yields an efficient and robust method for the test problems considered. Steps (2) and (4) each require $\mathcal{O}(mn)$ flops, and step (3) requires $\mathcal{O}(n)$ flops. A more in depth analysis reveals that construction of the subgradient vector requires $2mn$ flops when the current iterate is infeasible, and $(6m + 2)n$ flops when it is feasible. Here we have assumed that a sequential search of a length n array requires n flops. Referring to the discussion at the end of §4.4, computing the $(m-1) \times (m-1)$ Cholesky factors $\mathbf{L}^{(k+1)}$ and $\mathbf{\Lambda}^{(k+1)}$ requires $3(m-1)^2 + \mathcal{O}(m)$ flops and m square roots. Given that $m \ll n$ this work is negligible compared to the subgradient vector construction. Therefore, we conclude that each iteration requires $\mathcal{O}(n)$ flops. We note that these estimates apply to both the center-cut and deep-cut algorithms. The ellipsoid method also has the setup cost of computing $\mathbf{D}^{(0)} = \beta^2 r^2 (\hat{\mathbf{A}}^\top \hat{\mathbf{A}})^{-1}$. For window sizes $m \leq 4$ there exist analytic formulas for the inverse that require 32 flops when $m = 4$ and 7 flops when $m = 3$. In general, for $m > 4$ computing the inverse requires $(8/3)m^3$ flops to leading order. Moreover, computing the Cholesky factors of $\mathbf{D}^{(0)}$ requires $(m-1)^3/3$ flops to leading order. The computation of r requires $(m + 1)n$ flops.

In addition to the ellipsoid method per iteration and setup costs discussed above, there are additional overhead costs to consider. We note that in general the overhead costs represent only a small part of the overall ellipsoid method cost per iterant recombination step. After each multigrid cycle it is necessary to update $\hat{\mathbf{A}}$ and $\hat{\mathbf{A}}^\top\hat{\mathbf{A}}$. To be as efficient as possible we would like to recycle as much of the existing elements in $\hat{\mathbf{A}}$ and $\hat{\mathbf{A}}^\top\hat{\mathbf{A}}$ as possible. Prior to updating $\hat{\mathbf{A}}$ we have

$$\hat{\mathbf{A}} = [\mathbf{r}_2 - \mathbf{r}_1, \dots, \mathbf{r}_m - \mathbf{r}_1]$$

where $\mathbf{r}_i = \mathbf{A}\mathbf{x}_i$ is the ith residual for $i = 1, \dots, m$ (see (4.14)). Let

$$\hat{\mathbf{A}}_{old} = [\mathbf{r}_3 - \mathbf{r}_1, \dots, \mathbf{r}_m - \mathbf{r}_1],$$

where we have dropped the first column of $\hat{\mathbf{A}}$, and suppose that $\mathbf{r}_{new}$ is the new rightmost column of $\mathbf{AX}$. Then

$$\hat{\mathbf{A}}_{new} = [\hat{\mathbf{A}}_{old} + \mathbf{u}\mathbf{1}^\top \,|\, \mathbf{v}], \quad \mathbf{u} = \mathbf{r}_1 - \mathbf{r}_2, \quad \mathbf{v} = \mathbf{r}_{new} - \mathbf{r}_2. \tag{4.36}$$

Thus, by recycling $\hat{\mathbf{A}}_{old}$ we require only mn flops to update $\hat{\mathbf{A}}$. Now consider

$$\hat{\mathbf{A}}_{new}{}^\top\hat{\mathbf{A}}_{new} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix},$$

where

$$\begin{aligned}
\mathbf{H}_{11} &= \hat{\mathbf{A}}_{old}{}^\top\hat{\mathbf{A}}_{old} + (\mathbf{1}\mathbf{u}^\top\hat{\mathbf{A}}_{old})^\top + \mathbf{1}\mathbf{u}^\top\hat{\mathbf{A}}_{old} + \langle\mathbf{u},\, \mathbf{u}\rangle\mathbf{1}\mathbf{1}^\top,\\
\mathbf{H}_{21} &= \mathbf{v}^\top(\hat{\mathbf{A}}_{old} + \mathbf{u}\mathbf{1}^\top),\\
\mathbf{H}_{12} &= \mathbf{H}_{12}^\top,\\
\mathbf{H}_{22} &= \langle\mathbf{v},\, \mathbf{v}\rangle.
\end{aligned}$$

By recycling the $(m-2)\times(m-2)$ matrix $\hat{\mathbf{A}}_{old}{}^\top\hat{\mathbf{A}}_{old}$, computing the matrix $\mathbf{H}_{11}$ requires only $2mn + 3(m-2)^2 - 4n$ flops to leading order. Since $\mathbf{v}^\top\hat{\mathbf{A}}_{new} = [\mathbf{H}_{21} \,|\, \mathbf{H}_{22}]$, computing

$\mathbf{H}_{12}$, $\mathbf{H}_{21}$, and $\mathbf{H}_{22}$ requires $2n(m-1)$ flops to leading order. Thus, updating $\hat{\mathbf{A}}_{new}$ and $\hat{\mathbf{A}}_{new}{}^{\top}\hat{\mathbf{A}}_{new}$ requires only $4mn + 3(m-2)^2 - 6n$ flops to leading order.

4.6 One-norm minimization via linear programming

When $F(\mathbf{u}) = \|\mathbf{Au}\|_1$, the one-norm minimization problem (4.12) is formally equivalent to a *linear programming problem.* Linear programming is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints. By introducing the auxiliary variables

$$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^{\top}$$

the absolute values in the objective function can be eliminated, resulting in the equivalent linear program

$$\begin{aligned} &\text{minimize} \quad \mathbf{1}^{\top}\boldsymbol{\theta} \\ &\text{subject to} \quad \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{AX} & \mathbf{I} \\ -\mathbf{AX} & \mathbf{I} \\ \mathbf{1}^{\top} & \mathbf{0}^{\top} \\ -\mathbf{1}^{\top} & \mathbf{0}^{\top} \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \boldsymbol{\theta} \end{bmatrix} \geq \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ 1 \\ -1 \end{bmatrix} \\ &\qquad\qquad \mathbf{z} \in \mathbb{R}^m \quad \text{and} \quad \boldsymbol{\theta} \in \mathbb{R}^n, \end{aligned} \tag{4.37}$$

where the matrix describing the inequality constraints has $3n+2$ rows, $m+n$ columns, and $3nm + 2(n+m)$ nonzero elements. Standard methods for solving linear programs include the well-known *simplex method* as well as various *interior point methods* [12, 94]. The feasible set of any linear program can be represented as a *polyhedron*, that is, a set of the form

$$\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Cx} \geq \mathbf{b}\}.$$

An *extreme point* of a polyhedron $\mathcal{P}$ is any point in $\mathcal{P}$ that cannot be written as the convex combination of two other points in $\mathcal{P}$. A result from linear optimization states that if a linear program over a polyhedron $\mathcal{P}$ has an optimal solution, and if $\mathcal{P}$ has at least one extreme point, then the optimal solution must occur at an extreme point of $\mathcal{P}$ [12]. The simplex method searches for an optimal solution by moving along the boundary of $\mathcal{P}$ from one extreme point to another, always in a direction that reduces the value of the objective function. In this respect, the simplex method either locates the global optimum, or determines that no such optimum exists, that is, that the objective function is unbounded below on $\mathcal{P}$. Contrary to the simplex method, interior point methods traverse the interior of the feasible set to find an optimal solution to within some tolerance $\varepsilon > 0$. Essentially, interior point methods transform the original constrained problem into an unconstrained problem with an objective function that penalizes the boundary. This unconstrained problem is then approximately solved via Newton iteration. Interior point methods are particularly effective for large sparse problems, and often outperform the simplex method for these types of problems [12]. Although it is beyond the scope of this thesis to discuss these methods in detail, they present a viable alternative to the ellipsoid method discussed above, and should be investigated as part of future research for the iterant recombination one-norm minimization problem.

4.7 Numerical results

In this section we present the results of numerical tests for one-norm and two-norm minimization. We consider a subset of the test problems described in Chapter 3 including the tandem queueing network, the random walk on an unstructured directed planar graph, and the stochastic Petri net problem. All experiments are performed using Matlab version 7.5.0.342 (R2007b), where every attempt has been made to obtain optimized performance by exploiting sparse data types and vectorization in Matlab, and by implementing MEX (Matlab Executables) in the C programming language for the bottleneck operations in the multilevel methods. Timings are reported for a laptop running Windows XP, with a 2.50 GHz Intel Core 2 Duo processor and 4 GB of RAM. For the stopping criterion we iterate until the one-norm of the residual has been reduced by a factor of 10^{12}, that is, until

$$\frac{\|\mathbf{A}\mathbf{x}^{(k)}\|_1}{\|\mathbf{x}^{(k)}\|_1} < 10^{-12}\|\mathbf{A}\mathbf{x}^{(0)}\|_1. \tag{4.38}$$

The initial guess $\mathbf{x}^{(0)}$ is randomly generated by sampling the standard uniform distribution and then normalizing with respect to the one-norm.

The parameters for the multilevel aggregation method described in §4.1 are given in Table 4.1. We experiment with $\mathrm{F}(2,2)$-cycles and $\mathrm{W}(2,2)$-cycles, where the aggregates are frozen after ten iterations. In the tables below AGG-F denotes the F-cycle method and AGG-W denotes the W-cycle method.

Parameter	Value
Strength of connection parameter θ	0.25
Weighted Jacobi relaxation parameter ω	0.7
Maximum number of points on coarsest level	20
Maximum number of levels	20

Table 4.1: Parameters for AGG method.

Numerical tests have shown that iterant recombination with small window sizes is sufficient to dramatically reduce the overall number of multilevel iterations. Moreover, the constrained iterant recombination approach becomes less practical for larger window sizes owing to the increased computational cost of solving the corresponding optimization problem. Therefore, we consider AGG accelerated by iterant recombination with window sizes $m = 2, 3, 4$. To determine a reasonable stopping tolerance for the ellipsoid method (ε in (4.29)) we aim to strike a balance between the number of rejected iterates (step 7 in Algorithm 4.3) and the number of ellipsoid iterations. Let $\mathbf{r}^{(k)}$ be the residual of the kth multilevel iterate, let $\tilde{\mathbf{r}}$ be the residual of the computed solution from the iterant recombination procedure, and let $\mathbf{r}^\star$ be the residual of the exact solution to the iterant recombination optimization problem. Then $\|\mathbf{r}^\star\|_1 \leq \|\tilde{\mathbf{r}}\|_1$, and by the stopping criterion for the ellipsoid method

$$\|\mathbf{r}^{(k)}\|_1 - \|\mathbf{r}^\star\|_1 < \varepsilon.$$

Combining these inequalities we find that $\|\mathbf{r}^{(k)}\|_1 - \|\tilde{\mathbf{r}}\|_1 < \varepsilon$, or equivalently that

$$\frac{\|\tilde{\mathbf{r}}\|_1}{\|\mathbf{r}^{(k)}\|_1} < 1 + \frac{\varepsilon}{\|\mathbf{r}^{(k)}\|_1}. \tag{4.39}$$

In order to avoid rejection of the accelerated iterate we require the left-hand side of (4.39) be less than one. Therefore, we choose ε so that the right-hand side of this inequality is close to one. Since $\|\mathbf{r}^{(k)}\|_1 < \tau$ at convergence, where τ is the stopping tolerance of the multilevel method, one possibility for ε is to use a fixed stopping tolerance given by $\varepsilon_f = 0.01\tau$. However, experience has shown us that using a fixed tolerance results in a large number of ellipsoid method iterations during the initial multilevel cycles primarily because the multilevel iterates in $\mathbf{X}$ still have large residuals relative to τ. With respect to execution time it usually does not pay to solve the iterant recombination problem to a high degree of accuracy during these initial iterations. An alternative to the fixed stopping tolerance ε_f is a dynamic tolerance such as $\varepsilon_d = 0.01\|\mathbf{r}^{(k)}\|_1$. We note that as the multilevel method nears convergence ε_d approaches ε_f. As a point of comparison we consider the fixed and dynamic approaches in our numerical tests. If the stopping tolerance is not met within 400 iterations, the ellipsoid method is terminated. With respect to the two-norm minimization with `quadprog`, Matlab's default settings are used. In the case of window size two the exact solution is computed via the procedure described in §4.3.

In the tables below we report the total number of iterations required by the AGG algorithm accelerated with iterant recombination to converge for window size m. For the standalone AGG method we report the problem size on the finest level (n) the number of levels (levs) the number of iterations (it), and the operator complexity on the last cycle (C_{op}). In what follows we refer to the iterant recombination acceleration with two-norm minimization as two-norm acceleration, and in the one-norm case we say one-norm acceleration.

4.7.1 Tandem queueing network

Numerical results for the tandem queueing network are given in Table 4.2. It is evident that acceleration by iterant recombination leads to a large reduction in the number of AGG iterations. For $n = 262144$, the iteration counts are reduced by at least a factor of 3.7 for W-cycles, and by at least a factor of 5.8 for F-cycles. In general, iteration counts tend to decrease as the window size increases. Moreover, based on the iteration counts there does not appear to be any significant advantage of minimizing in the one-norm compared with minimizing in the two-norm. Although these results do not display perfect scalability (iteration counts grow as a function of problem size), the improvement in iteration counts is significant, and the scalability is much improved over the unaccelerated method.

Method	n	levs	C_{op}	it	Ellipsoid (ε_f)			Ellipsoid (ε_d)			Quadprog		
					Window size			Window size			Window size		
					2	3	4	2	3	4	2	3	4
AGG-W	1024	3	1.42	126	73*	47	42	66*	45	40	80	53	48
	4096	4	1.50	227	88	67	55	110	62	53	74*	97	64
	16384	5	1.50	286	125	84	61	114	74	62	86*	82*	71*
	65536	5	1.50	371	122	110	75	128*	92	78	108*	77*	79*
	262144	6	1.50	541	126	126	96	147*	114	99	109*	116*	116*
AGG-F	1024	3	1.42	126	73*	47	42	66*	45	40	80	53	48
	4096	4	1.47	273	94	75	62	122	73	59	68*	110	71
	16384	5	1.47	443	103	93	81	113*	98	73	92*	92*	86*
	65536	5	1.47	599	164	135	85	137*	113	88	110*	101*	96*
	262144	6	1.46	1085	186	153	124	182*	156*	170*	159*	164*	166*

Table 4.2: Tandem queueing network. Iteration counts to reduce the residual by a factor of 10^{12} for various window sizes and minimization strategies. The number of levels on the last cycle (levs), the operator complexity on the last cycle (C_{op}), and the number of iterations (it) are given for the unaccelerated AGG method. A superscript asterisk indicates there was at least one acceleration step in which the residual was not reduced, or in which the underlying optimization method failed to converge.

Figure 4.3 shows the convergence histories of the accelerated AGG W-cycles for $n = 262144$. Although our implementation of the iterant recombination procedure is not fully optimized, window size two acceleration appears to be the most efficient, reducing the execution time roughly by a factor of two. In Table 4.2, the ellipsoid method with the fixed stopping tolerance typically results in a better reduction of the iteration counts than the ellipsoid method with the dynamic stopping tolerance. However, as Figure 4.3 demonstrates, lower overall execution times are obtained with the dynamic stopping tolerance.

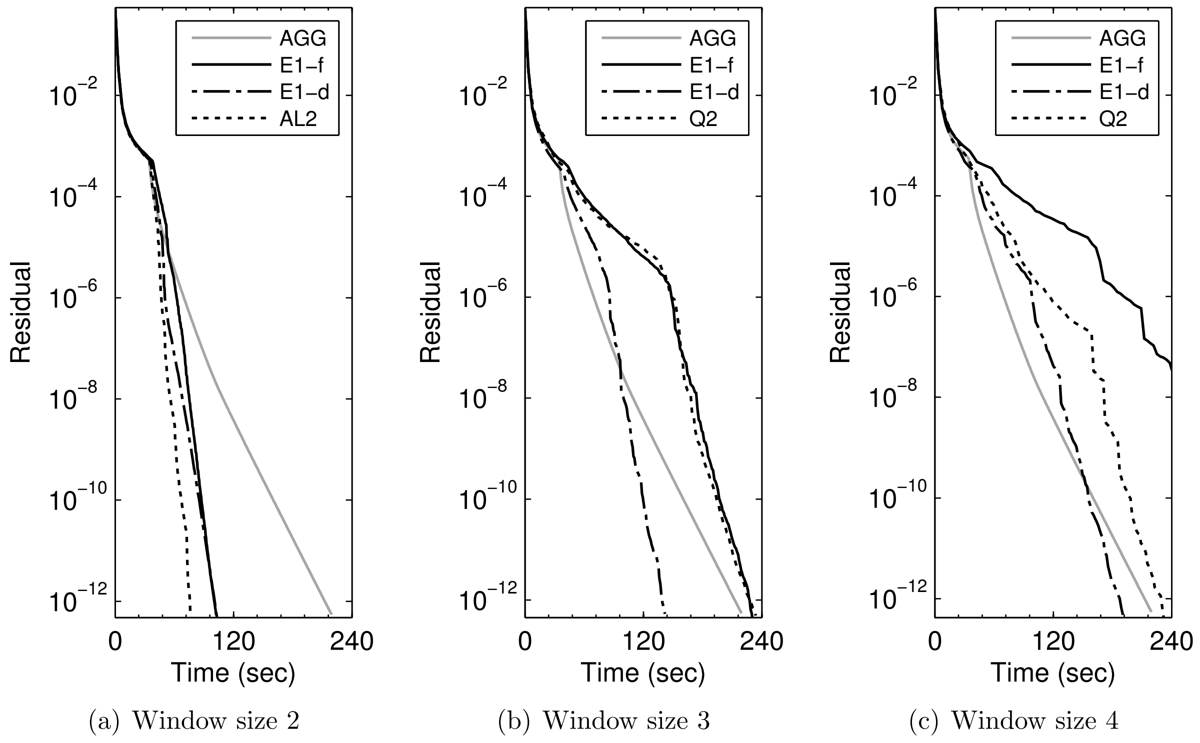
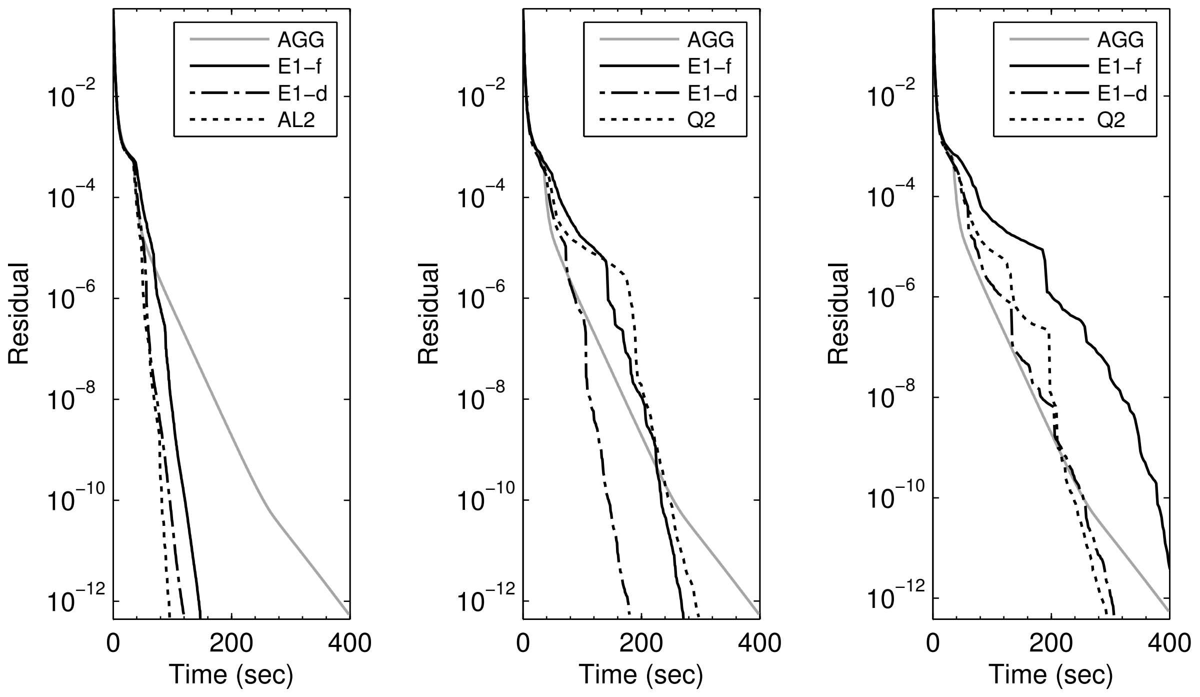


Figure 4.3: Convergence histories of AGG W-cycles for the tandem queueing network with $n = 262144$. In the legends AGG is the unaccelerated method, E1-f is one-norm minimization by the ellipsoid method with the fixed stopping tolerance, E1-d is one-norm minimization by the ellipsoid method with the dynamic stopping tolerance, Q2 is two-norm minimization by Matlab's `quadprog` method, and AL2 is the two-norm minimization method discussed in §4.3.

Similar convergence results are observed for the accelerated AGG F-cycles shown in Figure 4.4. Comparison with Figure 4.3 shows that for window size two, accelerated W-cycles are faster than accelerated F-cycles by a small margin. As the window size increases, this margin widens and the accelerated W-cycles are clearly faster, in particular those corresponding to one-norm minimization by the ellipsoid method with the dynamic stopping tolerance.



(a) Window size 2 (b) Window size 3 (c) Window size 4

Figure 4.4: Convergence histories of AGG F-cycles for the tandem queueing network with $n = 262144$. In the legends AGG is the unaccelerated method, E1-f is one-norm minimization by the ellipsoid method with the fixed stopping tolerance, E1-d is one-norm minimization by the ellipsoid method with the dynamic stopping tolerance, Q2 is two-norm minimization by Matlab's `quadprog` method, and AL2 is the two-norm minimization method discussed in §4.3.

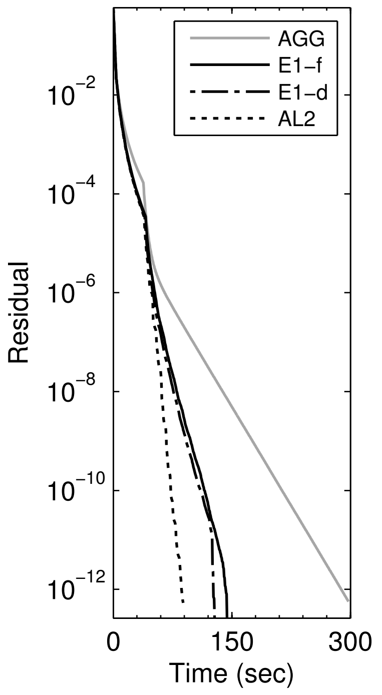
4.7.2 Unstructured planar graph

Numerical results for the unstructured directed planar graph are given in Table 4.3. For $n = 262144$ the iteration counts are reduced by at least a factor of 3.3 for both W-cycles and F-cycles, and by at least a factor of 3.3 for F-cycles. Two-norm minimization with window size two appears to be the most efficient, and there is not a significant difference in terms of iteration counts between two-norm minimization with Matlab's `quadprog` and the one-norm minimization approach for window sizes greater than two.

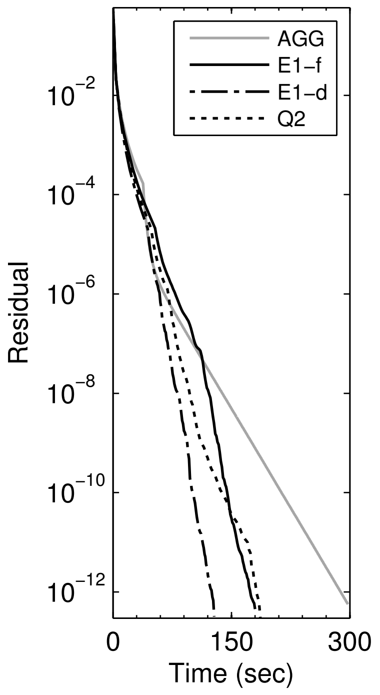
Figure 4.3 shows the convergence histories of the accelerated AGG W-cycles for $n = 262144$. Again, two-norm minimization with window size two is the most efficient acceleration method, reducing the execution time roughly by a factor of three. As observed for the previous test problem, one-norm acceleration is more efficient with the dynamic stopping tolerance than with the fixed stopping tolerance.

Method	n	levs	C_{op}	it	Ellipsoid (ε_f)			Ellipsoid (ε_d)			Quadprog		
					Window size			Window size			Window size		
					2	3	4	2	3	4	2	3	4
AGG-W	1024	3	1.29	122	52	44	36	52*	46	34	66	48	42
	4096	4	1.34	182	85	58	47	95	54	41	71*	69	59
	16384	5	1.36	299	127*	66	55	128	64	53	81*	60*	60*
	65536	5	1.37	457	159	82	72	138*	67	66	98*	72*	66*
	262144	6	1.37	560	170*	95	87	155*	86	76	100*	97*	76*
AGG-F	1024	3	1.29	122	52	44	36	52*	46	34	66	48	42
	4096	4	1.32	194	92*	59	54	103	53	47	56*	73	64
	16384	5	1.33	373	133	100	72	130*	70	63	94*	76*	69*
	65536	6	1.34	594	195*	83	78	136*	93	75	103*	83*	81*
	262144	6	1.34	903	277	136	109	180*	106	98	126*	150*	99*

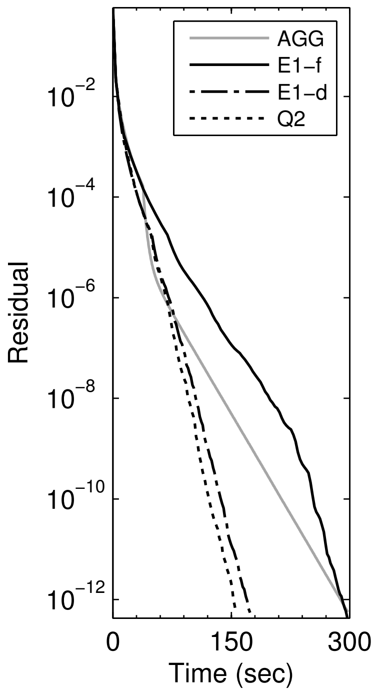
Table 4.3: Unstructured directed planar graph. Iteration counts to reduce the residual by a factor of 10^{12} for various window sizes and minimization strategies. The number of levels on the last cycle (levs), the operator complexity on the last cycle (C_{op}), and the number of iterations (it) are given for the unaccelerated AGG method. A superscript asterisk indicates there was at least one acceleration step in which the residual was not reduced, or in which the underlying optimization method failed to converge.



(a) Window size 2



(b) Window size 3



(c) Window size 4

Figure 4.5: Convergence histories of AGG W-cycles for the unstructured directed planar graph with $n = 262144$. In the legends AGG is the unaccelerated method, E1-f is one-norm minimization by the ellipsoid method with the fixed stopping tolerance, E1-d is one-norm minimization by the ellipsoid method with the dynamic stopping tolerance, Q2 is two-norm minimization by Matlab's `quadprog` method, and AL2 is the two-norm minimization method discussed in §4.3.

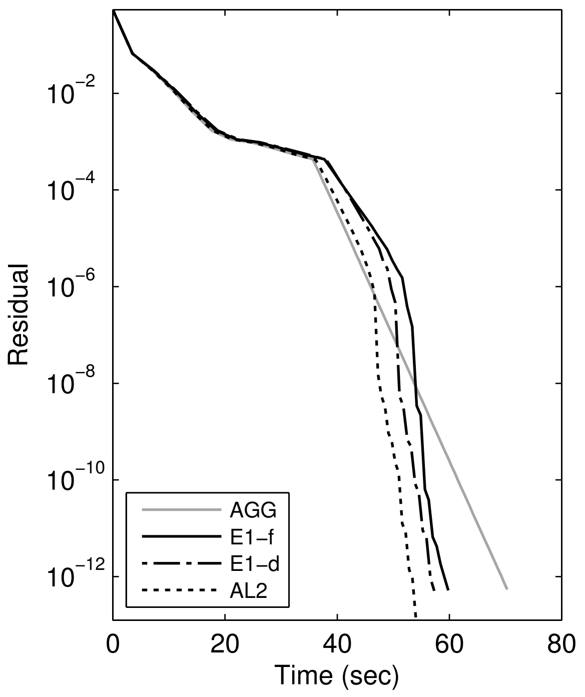
4.7.3 Stochastic Petri net

Numerical results for the stochastic Petri net are given in Table 4.4. We note that the test case with $n = 1015$ was generated with the initial marking $(13, 0, 0, 0, 0)$. It appears that accelerated W-cycles with window size two are the most efficient, displaying near-optimal performance. In contrast to the previous test problems the unaccelerated AGG W-cycles also display scalable performance. While the iterant recombination procedure is still able to reduce the iteration counts in this case, the reduction is less significant compared with the previous test problems. For $n = 255346$ the iteration counts are reduced by at least a factor of 1.5 for W-cycles, and by at least a factor of 1.7 for F-cycles. This test problem is meant to illustrate how iterant recombination acceleration typically improves methods that are far from being optimal, thus improving robustness, but is less effective for methods and problems that already display near-optimal convergence.

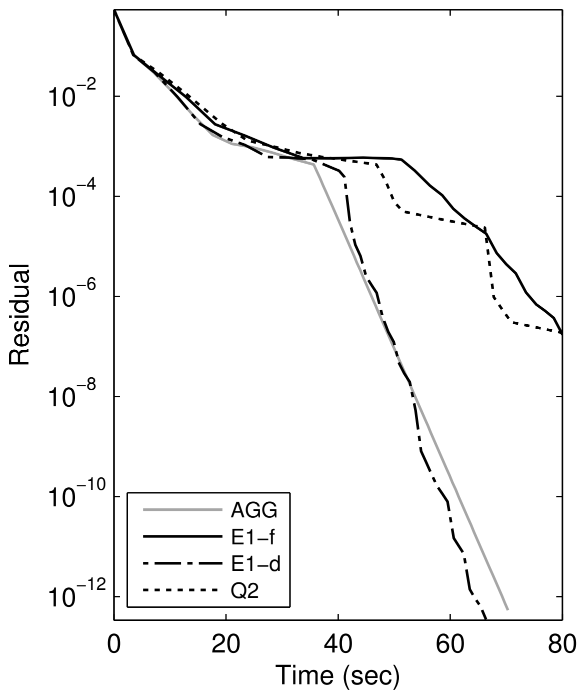
Method	n	levs	C_{op}	it	Ellipsoid (ε_f)			Ellipsoid (ε_d)			Quadprog		
					Window size			Window size			Window size		
					2	3	4	2	3	4	2	3	4
AGG-W	1015	4	1.77	74	29	28	23	34	27	24	31*	29*	26*
	4324	5	1.87	72	30	30	30	30	31	30	32*	36*	28*
	16206	6	1.93	72	31	37	32	35	35	35	32*	41*	33*
	60116	6	1.91	73	36	38	38	34	38*	38*	35*	39*	39*
	255346	7	1.70	73	37	49	48	37	38*	45	39*	32*	39*
AGG-F	1015	4	1.71	80	33	27	27	36	28	25	33*	25*	25*
	4324	5	1.78	90	34	38	33	36*	29	29	34*	32*	31*
	16206	6	1.81	89	39	39	37	39	42	41	42*	41*	37*
	60116	6	1.66	126	49	51	44	50	61*	43*	46*	47*	48*
	255346	6	1.51	125	50	69	56	74	67*	63	54*	48*	69*

Table 4.4: Stochastic Petri net. Iteration counts to reduce the residual by a factor of 10^{12} for various window sizes and minimization strategies. The number of levels on the last cycle (levs), the operator complexity on the last cycle (C_{op}), and the number of iterations (it) are given for the unaccelerated AGG method. A superscript asterisk indicates that there was at least one acceleration step in which the residual was not reduced, or in which the underlying optimization method failed to converge.

Figure 4.6 shows the convergence histories of accelerated AGG W-cycles for $n = 255346$. We note that timing results for window size four are not shown because compared with the standalone AGG method, acceleration with window size four resulted in slower overall execution times. In this case iterant recombination acceleration results in only modest speedups. The most efficient acceleration method is two-norm minimization with window size two, which reduces the execution time approximately by a factor of 1.3.



(a) Window size 2



(b) Window size 3

Figure 4.6: Convergence histories of AGG W-cycles for the stochastic Petri net with $n = 255346$. In the legends AGG is the unaccelerated method, E1-f is one-norm minimization by the ellipsoid method with the fixed stopping tolerance, E1-d is one-norm minimization by the ellipsoid method with the dynamic stopping tolerance, Q2 is two-norm minimization by Matlab's `quadprog` method, and AL2 is the two-norm minimization method discussed in §4.3.

4.8 General discussion and conclusions

The main contribution of this chapter was to show that adaptive multilevel aggregation for Markov chains can be accelerated through constrained iterant recombination on the finest level. Numerical results demonstrated that when the AGG method is not performing optimally significant improvements in its iteration counts and scalability are possible by iterant recombination with small window sizes. Moreover, when the AGG method was performing optimally, iterant recombination was able to further reduce iteration counts and execution times, although not to the same degree as in the non-optimal test problems. For window sizes larger than three the iteration counts were not further reduced by a significant amount, and in most cases the added overhead of solving a larger minimization problem (more unknowns) lead to increased execution times. In terms of execution time, the ellipsoid method was typically faster than or as fast as `quadprog` for window sizes greater than two. However, in terms of overall execution time, window size two acceleration was the clear winner, with the analytic solution method discussed in §4.3 resulting in the fastest overall execution times for all test problems. While it is difficult to form generally valid conclusions based on timing results for one-norm and two-norm minimization due to possible differences in implementation efficiency, in terms of iteration reduction no significant difference was observed between the one-norm minimization and two-norm minimization approaches.

Chapter 5

Over-Correction for AMG Methods for Markov Chains

Simple non-overlapping multilevel aggregation has computational advantages over methods such as the MCAMG method discussed in Chapter 3 in terms of memory complexity and computational complexity per cycle (see §2.6.4). However, simple multilevel aggregation typically suffers from slow convergence due to the inability of the coarse-grid correction to effectively remove smooth error components. In the previous chapter we discussed a general approach based on iterant recombination to accelerate multilevel methods applied to Markov chain problems. Numerical results demonstrated that our approach can significantly improve the performance and scalability of simple multilevel aggregation for Markov chains. In this chapter we consider an altogether different approach to accelerate simple multilevel aggregation for Markov chains that is based on scaling the coarse-grid correction by a scalar α. Since α is taken to be greater than one, we use the terminology of Míka and Vaněk [125] and refer to this acceleration as multilevel aggregation for Markov chains with *over-correction*. In particular, we present an *automatic* over-correction mechanism, applicable on all levels, that can cheaply and effectively improve the convergence of the simple multilevel aggregation algorithm described in §4.1 (Algorithm 4.1). We compare our automatic mechanism with a fixed over-correction approach in which a suitable α is chosen a priori via trial and error. In addition, we compare multilevel aggregation accelerated by over-correction with unaccelerated multilevel aggregation, multilevel aggregation accelerated by iterant recombination from Chapter 4, the MCAMG method from Chapter 3, and leading Krylov subspace methods including the preconditioned stabilized biconjugate gradient (Bi-CGStab) method [104], and the preconditioned generalized minimal residual (GMRES) method [104].

We begin by briefly describing the classical over-correction mechanism for standard additive-correction multigrid applied to symmetric positive definite systems in §5.1. In §5.2 we describe our automatic over-correction mechanism for multiplicative-correction multilevel aggregation for Markov chains. In §5.3 we present numerical results, and §5.4 contains the concluding remarks.

5.1 Classical over-correction for multilevel aggregation

The idea of applying over-correction is a simple one, its goal being to improve the rate of convergence of multilevel aggregation methods [112, 125]. With respect to additive correction schemes, instead of applying a standard coarse-grid correction, an over-correction parameter α is introduced to scale the correction:

$$\mathbf{x}^{\text{CGC}} = \bar{\mathbf{x}} + \alpha \mathbf{P}\mathbf{e}_c, \tag{5.1}$$

where $\bar{\mathbf{x}}$ is the relaxed fine-level approximation and α is a positive scalar typically larger than one. Closely following the example of Stüben [112] (see also [13]), over-correction as in (5.1) can be motivated by considering a simple model problem:

$$\begin{aligned} \frac{d^2u}{dx^2} &= f(x) \quad \text{for} \quad x \in (0,1), \\ u(x) &= 0 \quad \text{for} \quad x \in \{0,1\}. \end{aligned} \tag{5.2}$$

Assuming the model problem (5.2) is discretized with uniform grid spacing h and centered finite difference approximations, we obtain a linear system $\mathbf{Au} = \mathbf{f}$ in which

$$\mathbf{A} = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

is a symmetric positive definite matrix. Now let $\mathbf{e}$ be any fine-level error that satisfies the boundary conditions in (5.2). Then by the variational principle of the Galerkin coarse-level system (see Theorem 2.6.1), the two-level correction $\mathbf{Pe}_c$ is optimal in the sense that it minimizes $\|\mathbf{e} - \mathbf{Pe}_c\|_\mathbf{A}$ with respect to all corrections in the range of $\mathbf{P}$. Consequently, $\mathbf{Pe}_c$ minimizes

$$\langle \mathbf{Av}, \mathbf{v} \rangle = \frac{1}{2h^2} \Big(\sum_i \sum_{j \in \mathcal{N}_i} (v_i - v_j)^2 + \sum_i s_i v_i^2 \Big), \quad s_i = 2 + \sum_{j \neq i} a_{ij}, \tag{5.3}$$

where $\mathbf{v} = \mathbf{e} - \mathbf{Pe}_c$ and $\mathcal{N}_i$ is the set of nearest neighbors of i. This result implies that away from the boundary where $s_i = 0$, the two-norm of the slope of $\mathbf{v}$ is minimal, whereas at the boundary where $s_i \neq 0$ the corresponding components of $\mathbf{v}$ are zero. Now suppose that aggregation is defined by grouping pairs of neighboring variables together and consider a smooth error $\mathbf{e}$. The result of this minimization near the left boundary of the interval $[0, 1]$ is illustrated in Figure 5.1. Interpolation is constant over the aggregates, and the slope of $\mathbf{Pe}_c$ is zero. Between the aggregates, the slope of $\mathbf{v}$ becomes minimal if the slope of $\mathbf{Pe}_c$ equals the slope of $\mathbf{e}$. Consequently, the correction $\mathbf{Pe}_c$ has approximately half the slope of $\mathbf{e}$. As illustrated in Figure 5.1, multiplying $\mathbf{Pe}_c$ by a factor of two gives a much more effective correction with respect to the error $\mathbf{e}$.

One approach for over-correction is to use a fixed value of α that is determined by either a theoretical analysis, or through an a priori trial-and-error strategy [13, 17]. The main challenge, however, is to *automatically* determine an appropriate value for α. In [125] a method for automatically determining α was suggested by minimizing the energy norm

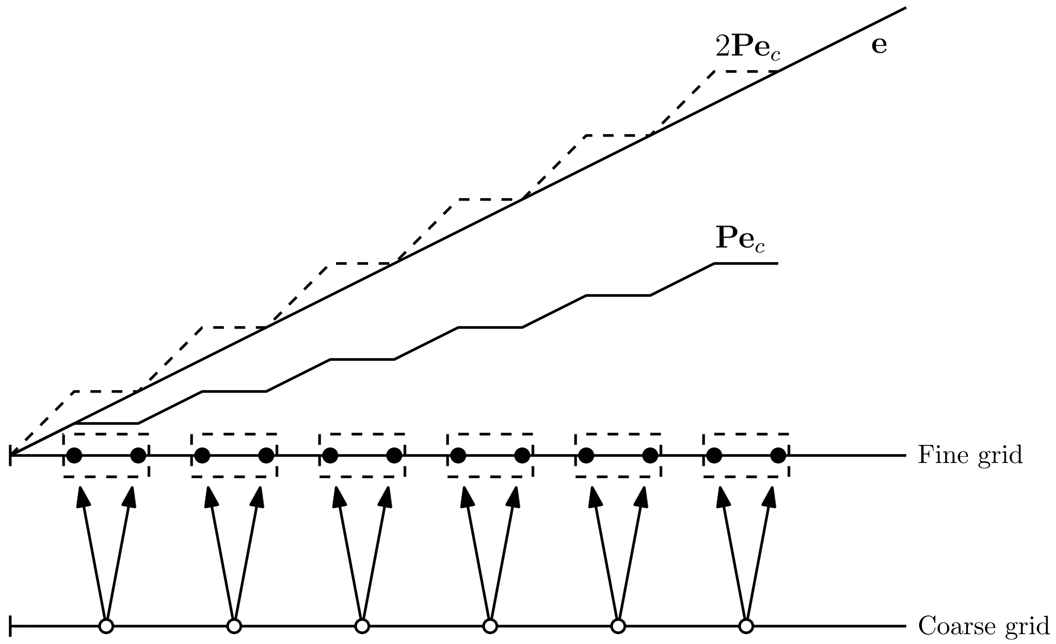


Figure 5.1: Optimal approximation $\mathbf{Pe}_c$ of the smooth error $\mathbf{e}$ with respect to the energy norm for the model problem. Dashed boxes represent the aggregates on the fine grid, solid dots correspond fine-grid points, and hollow dots correspond to coarse-grid points.

of the error corresponding to the coarse-grid correction:

$$\text{minimize} \quad \left\| \mathbf{H}^{\nu_2}(\mathbf{e} - \alpha \mathbf{Pe}_c) \right\|_{\mathbf{A}}^2 \quad \text{over all} \quad \alpha \in \mathbb{R}, \tag{5.4}$$

where $\mathbf{e} = \mathbf{x} - \bar{\mathbf{x}}$ is the unknown error of the smoothed iterate $\bar{\mathbf{x}}$ prior to coarse-grid correction, and $\mathbf{H}$ is the iteration matrix of the relaxation scheme. The error of the coarse-grid correction is smoothed by $\nu_2 > 0$ relaxations prior to computing α because the goal of (5.4) is to find the over-correction parameter that yields the optimal solution *after post-relaxation.* The optimal α can then be calculated by the formula:

$$\alpha_{opt} = \frac{\hat{\mathbf{e}}^\top(\mathbf{f} - \mathbf{A}\hat{\mathbf{x}})}{\hat{\mathbf{e}}^\top \mathbf{A}\hat{\mathbf{e}}} \quad \text{with} \quad \hat{\mathbf{e}} = \mathbf{H}^{\nu_2}\mathbf{e} \quad \text{and} \quad \hat{\mathbf{x}} = \text{Relax}(\mathbf{A}, \mathbf{f}, \bar{\mathbf{x}}, \nu_2). \tag{5.5}$$

Although this approach has been demonstrated to significantly improve the performance of simple multilevel aggregation applied to symmetric positive definite systems, it is limited by the requirement that $\mathbf{A}$ is a symmetric positive definite matrix. If $\mathbf{A}$ is not symmetric positive definite then $\|\cdot\|_{\mathbf{A}}$ does not define a norm, and it becomes unclear how to minimize the quantity $\mathbf{e} - \alpha\mathbf{P}\mathbf{e}_c$ given that $\mathbf{e}$ is the *unknown* error. Moreover, a large number of pre-relaxation steps may be necessary in order for the effect of over-correction to be significant, for example, see [122].

5.2 Over-Correction for Markov chains

For a nonsymmetric matrix $\mathbf{A}$, the expression (5.4) is no longer meaningful because the energy norm is undefined. As far as we are aware, only fixed over-correction strategies have been considered for nonsymmetric problems in the literature [27, 62, 130, 131]. In particular, Horton and Leutenegger [69] experimented with a fixed over-correction approach for Markov chains by taking a convex combination of two different coarse approximations to the correction. However, their approach was not automated. In order to automatically determine α, we consider minimizing the residual two-norm as an alternative to the energy norm. We note that minimization of the residual two-norm to accelerate the convergence of multilevel processes has already been considered [129], but only on the finest level.

The multilevel aggregation method for Markov chains described in §4.1 employs a multiplicative coarse-grid correction of the form

$$\mathbf{x}^{\text{CGC}} = \mathbf{P}\mathbf{x}_c = \text{diag}(\bar{\mathbf{x}}^{(k)})\mathbf{Q}\,\text{diag}(\mathbf{Q}^\top\bar{\mathbf{x}}^{(k)})^{-1}\mathbf{x}_c,$$

where $\mathbf{x}_c$ is the solution of the coarse-level problem. Letting $\bar{\mathbf{P}} = \text{diag}(\bar{\mathbf{x}}^{(k)})\mathbf{Q}$ and $\mathbf{e}_c = \text{diag}(\mathbf{Q}^\top\bar{\mathbf{x}}^{(k)})^{-1}\mathbf{x}_c$, the coarse-grid correction can be written as

$$\mathbf{x}^{\text{CGC}} = \bar{\mathbf{P}}\mathbf{e}_c,$$

where $\mathbf{e}_c$ is the coarse-level approximation of the fine-level multiplicative error. Owing to the nature of the coarse-grid correction, we cannot simply multiply the correction $\mathbf{Q}\mathbf{e}_c$ by a scalar α, as doing so would result in α being eliminated by normalization on the fine level. Instead, we consider a coarse-grid correction of the form

$$\mathbf{x}_{\alpha}^{\text{CGC}} = \operatorname{diag}(\bar{\mathbf{x}}^{(k)})(\mathbf{Q}\mathbf{e}_c)^{\alpha}, \tag{5.6}$$

where the αth power is applied componentwise to the multiplicative correction. Note that $\mathbf{Q}\mathbf{e}_c$ is the fine-level approximation of the fine-level multiplicative error. Taking the componentwise logarithm of (5.6) we observe that

$$\log((\mathbf{x}_{\alpha}^{\text{CGC}})_i) = \log(\bar{x}_i^{(k)}) + \alpha \log((\mathbf{Q}\mathbf{e}_c)_i) \quad \text{for each} \quad i,$$

which is in some sense analogous to the additive over-correction formula (5.1). In particular, when the multiplicative error component $(\mathbf{Q}\mathbf{e}_c)_i = 1$, the over-correction is inactive. Since the disaggregation matrix $\mathbf{Q}$ corresponds to piecewise constant interpolation, (5.6) is equivalent to

$$\mathbf{x}_{\alpha}^{\text{CGC}} = \bar{\mathbf{P}}(\mathbf{e}_c)^{\alpha}. \tag{5.7}$$

The goal then is to find the value of α that minimizes

$$\left\| \mathbf{A}\bar{\mathbf{P}}(\mathbf{e}_c)^{\alpha} \right\|_2^2. \tag{5.8}$$

Unfortunately, minimization of (5.8) is difficult as well as computationally expensive due to the nonlinearity of the correction. Moreover, we observe that if all the components of $\mathbf{e}_c$ are greater than one (resp. less than one) then the optimal over-correction parameter is $\alpha = -\infty$ (resp. $\alpha = \infty$). As a workaround we use the fact that the multiplicative correction achieved by the multilevel aggregation method is typically close to $\mathbf{1}$, that is, we assume that

$$\mathbf{Q}\mathbf{e}_c = \mathbf{1} + \boldsymbol{\varepsilon} \quad \text{for some vector } \boldsymbol{\varepsilon} \text{ such that} \quad \|\boldsymbol{\varepsilon}\|_\infty \ll 1.$$

By the componentwise application of Taylor's theorem

$$(\mathbf{Q}\mathbf{e}_c)^\alpha = (\mathbf{1} + \boldsymbol{\varepsilon})^\alpha = \mathbf{1} + \alpha\boldsymbol{\varepsilon} + \mathcal{O}(\boldsymbol{\varepsilon}^2). \tag{5.9}$$

Therefore, a linearized over-correction formula is given by

$$\operatorname{diag}(\bar{\mathbf{x}}^{(k)})(\mathbf{Q}\mathbf{e}_c)^\alpha \approx \bar{\mathbf{x}}^{(k)} + \alpha \operatorname{diag}(\bar{\mathbf{x}}^{(k)})\boldsymbol{\varepsilon} = \bar{\mathbf{x}}^{(k)} + \alpha(\mathbf{x}^{\text{CGC}} - \bar{\mathbf{x}}^{(k)}). \tag{5.10}$$

We note that formula (5.10) is equivalent to the additive over-correction formula (5.1) with $\mathbf{P}\mathbf{e}_c = \mathbf{x}^{\text{CGC}} - \bar{\mathbf{x}}^{(k)}$. Similar to the additive correction scheme, in order to compute the optimal parameter α, the corrected approximation is first smoothed by $\nu > 0$ relaxations:

$$\hat{\mathbf{x}} = \operatorname{Relax}(\mathbf{A}, \mathbf{0}, \mathbf{x}^{\text{CGC}}, \nu). \tag{5.11}$$

Although the variational principle of the Galerkin coarse-level system is not applicable in Markov chain applications, we have found that relaxing the corrected approximation $\mathbf{x}^{\text{CGC}}$ prior to computing α improves the performance of the over-correction process. In particular, relaxing $\mathbf{x}^{\text{CGC}}$ ensures that $\bar{\mathbf{x}}^{(k)}$ is not significantly smoother than $\hat{\mathbf{x}}$, which results in a better search direction $\hat{\mathbf{x}} - \bar{\mathbf{x}}^{(k)}$. The optimal parameter α is then obtained by minimizing

$$\left\|\mathbf{R}\mathbf{A}(\bar{\mathbf{x}}^{(k)} + \alpha(\hat{\mathbf{x}} - \bar{\mathbf{x}}^{(k)}))\right\|_2^2, \tag{5.12}$$

with the minimizer given by

$$\alpha_{opt} = \frac{\mathbf{u}^\top(\mathbf{u} - \mathbf{v})}{\langle \mathbf{u} - \mathbf{v}, \mathbf{u} - \mathbf{v}\rangle}, \quad \mathbf{u} = \mathbf{R}\mathbf{A}\bar{\mathbf{P}}\mathbf{1}_c, \quad \mathbf{v} = \mathbf{Q}^\top\mathbf{A}\hat{\mathbf{x}}. \tag{5.13}$$

We note that restricting the residual of the linearized over-correction to the coarse level as in (5.12) improves convergence and reduces the computational cost of computing α_{opt}. Restriction has a similar effect as smoothing in that it helps expose algebraically smooth error components, but is much cheaper to apply than a relaxation.

The amount of computational work to compute α_{opt} on the ℓth level of a multigrid

V-cycle is approximately

$$(\nu+1)\,\mathrm{nnz}(\mathbf{A}_\ell)+3n_\ell+\mathrm{nnz}(\mathbf{A}_{\ell+1})+5n_{\ell+1} \tag{5.14}$$

flops to leading order. The dominant computation in (5.13) is computing the vector $\mathbf{v}$ which includes the cost of the ν additional relaxations. The computational cost of automatic over-correction for a V-cycle ($\ell=0,\ldots,L-1$) is then approximately

$$((\nu+2)C_{op}-1)\,\mathrm{nnz}(\mathbf{A}_0)+(8C_{grid}-5)n_0 \tag{5.15}$$

flops to leading order, which is roughly the cost of $(\nu+2)C_{op}-1$ relaxations on the finest level. The overall computational cost for a W-cycle is more complicated to compute because coarser levels are visited more than once, but it is bounded above and below by $(\nu+2)C_{op}+8C_{grid}$ flops and $(\nu+1)C_{op}+3C_{grid}$ flops, respectively.

In practice we restrict α_{opt} to a predefined interval $[\alpha_{min},\alpha_{max}]$ with $0<\alpha_{min}<\alpha_{max}$ to obtain a more stable algorithm. For example, in some cases formula (5.13) may yield a negative value for α_{opt}, which could potentially cause numerical instability. If α_{opt} falls outside the interval $[\alpha_{min},\alpha_{max}]$, then it is set to either α_{min} or α_{max}, depending on which boundary point it is nearest. In addition, it may also be necessary to use a smoothing parameter in (5.11) that is different from the parameter used in the pre- and post-relaxations, or, it may even be necessary to use $\nu>1$ relaxations to smooth the coarse-grid correction. In general, insufficient smoothing of the coarse-grid correction may result in a poor determination of α_{opt}.

Empirical evidence has shown us that using the linearized form (5.10) of the coarse-grid over-correction equation in conjunction with automatic over-correction results in better overall performance of the multilevel method than applying α as a componentwise power as in (5.7). However, using the linearized formula may result in a coarse-grid correction that has nonpositive values. In order for the linearized over-correction (5.10) to have strictly positive components we require that

$$0 < \alpha \leq \inf_{i \in \mathcal{I}^-} \left(\frac{\bar{x}_i^{(k)}}{|\hat{x}_i - \bar{x}_i^{(k)}|} \right), \tag{5.16}$$

where $\mathcal{I}^- = \{i : \hat{x}_i - \bar{x}_i^{(k)} < 0\}$. Checking this condition on each level is computationally expensive, therefore we check the over-corrected coarse-grid correction for any negative components, and if any are found, $\mathbf{x}_\alpha^{\text{CGC}}$ is replaced by the relaxed coarse-grid correction $\hat{\mathbf{x}}$ in (5.11) and the post-relaxations are skipped. In our experience, nonpositive values only occur during the initial iterations when the residual norm is still relatively large and the computed solution is still far from the exact solution.

5.3 Numerical results

In this section we present the results of our numerical tests for the tandem queueing network, the random walk on an unstructured directed planar graph, the stochastic Petri net problem, and the octagonal mesh problem. The methods tested include weighted Jacobi relaxation, simple non-overlapping multilevel aggregation (AGG), AGG accelerated by over-correction (OC-AGG), AGG accelerated by iterant recombination (IR-AGG), the MCAMG method from Chapter 3, ILU-preconditioned stabilized biconjugate gradient (Bi-CGStab) [104], and ILU-preconditioned GMRES [104]. All experiments are performed using Matlab version 7.11.0.584 (R2010b) 64-bit, and every attempt has been made to obtain optimized performance by exploiting sparse data types and vectorization in Matlab, and by implementing MEX (Matlab Executables) files in the C programming language for the bottleneck operations in the multilevel methods. Timings are reported for server running Red Hat Enterprise Linux (release 5.8), with four 2.60 GHz Dual-Core AMD Opteron 2218 processors and 8 GB of RAM. We note that the difference in computing environments between this chapter and Chapters 3 and 4 is to facilitate larger test problems.

We use the following stopping criterion for all methods except GMRES:

$$\frac{\|\mathbf{A}\mathbf{x}^{(k)}\|_1}{\|\mathbf{x}^{(k)}\|_1} < 10^{-12}\|\mathbf{A}\mathbf{x}^{(0)}\|_1. \tag{5.17}$$

For GMRES we use

$$\left(\|\mathbf{A}\mathbf{x}^{(k)}\|_2 < 10^{-12}\|\mathbf{A}\mathbf{x}^{(0)}\|_1 \quad \text{and} \quad \frac{\|\mathbf{A}\mathbf{x}^{(k)}\|_1}{\|\mathbf{x}^{(k)}\|_1} < 10^{-12}\|\mathbf{A}\mathbf{x}^{(0)}\|_1\right). \tag{5.18}$$

The initial guess $\mathbf{x}^{(0)}$ is randomly generated by sampling the standard uniform distribution and then normalizing with respect to the one-norm. In order to check the one-norm criterion in (5.18) at each inner iteration of GMRES it is necessary to compute the current approximation $\mathbf{x}^{(k)}$ (see [104] for details). However, doing so may incur unnecessary extra computations, especially when GMRES is still far from converging. Therefore, we use a preliminary stopping criterion based on the two-norm of the residual, which we get for free (see [104]), to gauge if GMRES is near convergence, and only when this condition is satisfied do we compute $\mathbf{x}^{(k)}$ and check the one-norm criterion in (5.18). We note that although the iterates $\mathbf{x}^{(k)}$ are not normalized in the inner GMRES iterations, their one-norms remain close to one, and so the two-norm test is meaningful.

The Bi-CGStab and GMRES algorithms were implemented according to the templates in [5]. Experiments with GMRES were run for subspaces with dimensions 10, 25, and 50. We also considered the following preconditioners: ILU(0) and ILUTP (ILU with thresholding and pivoting [104]) with permutation tolerance set to 1 (always choose the maximum magnitude element in the column as the pivot) and with drop tolerances 0.1, 0.01, 0.001, and 0.0001. The preconditioners were constructed by Matlab's built-in sparse incomplete LU factorization method `ilu`. For each test problem we selected the Bi-CGStab and GMRES parameters that gave fastest execution time for the largest problem size, and used that combination for all problem sizes. We note that in some instances very small non-positive components were present in the computed solution, in which case we took the absolute value and renormalized. A run was terminated if it reached the iteration limit of 500 iterations before satisfying the stopping criterion.

The parameters for the AGG, IR-AGG, and OC-AGG methods are given in Table 4.1. We use W$(2, 2)$-cycles and neighborhood aggregation with aggregates frozen on all levels after five iterations. A run was terminated if it reached the iteration limit of 1000 iterations before satisfying the stopping criterion. With respect to IR-AGG we choose between the fastest of the analytic solution method for two-norm minimization with window size two (IR-AGG-A2) and the one-norm minimization by the ellipsoid method with window size three (IR-AGG-E3). With respect to OC-AGG we denote the fixed over-correction method by OC-AGG-f and the automatic over-correction method by OC-AGG-a. In the case of OC-AGG-a we use $\nu = 2$ weighted Jacobi relaxations with $\omega = 0.7$ to smooth the correction and we restrict the computed value of α to the interval $[1.1, 2]$. The parameters for the MCAMG method are given in Table 3.1. We use V$(2, 2)$-cycles with transfer operators frozen after five iterations (unless specified otherwise). Experience shows that in general these settings work well for a wide range of problems.

In the tables below we report the problem size on the finest level (n), the number of iterations to converge (it), the overall execution time in seconds (time), and the operator complexity (C_{op}). The weighted Jacobi relaxation parameter reported in the tables is the relaxation parameter that minimizes its iteration count. In the case of GMRES and Bi-CGStab the overall execution times include the time to construct the preconditioner. In Blank entries in a table indicate that a method failed to satisfy its stopping criterion within its alloted iteration limit. We note that in one instance it was also necessary to terminate a run after a sufficiently long period of time (approximately two hours).

5.3.1 Tandem queueing network

Numerical results for the tandem queueing network are given in Tables 5.1 and 5.2. We observe that AGG with fixed over-correction parameter $\alpha = 2.2$ is the most efficient method for this test problem, displaying near-optimal scalability. Although OC-AGG-a is almost four times faster than unaccelerated multilevel aggregation, it is unable to obtain the same kind of scalable performance as OC-AGG-a. We observe that OC-AGG-A2 is also a competitive solver for this test problem with near-optimal scalability and the second fastest execution times for the largest test cases. Preconditioned GMRES and Bi-CGStab are competitive solvers for test cases with $n \leq 1048576$, but the multilevel methods are faster than Bi-CGStab for the larger problem sizes. Moreover, the execution time scalability of the multilevel methods is significantly better than that of Bi-CGStab and GMRES (Figure 5.2).

	Weighted Jacobi ($\omega = 0.99$)		Bi-CGStab (ILUTP(10^{-4}))		GMRES(50) (ILUTP(10^{-4}))		AGG		
n	it	time	it	time	it	time	it	C_{op}	time
4096	50287	7.1	5	< 0.1	8	< 0.1	227	1.50	1.6
16384	182441	150.6	7	0.6	12	0.6	286	1.50	6.8
65536			11	3.7	17	3.5	369	1.50	30.3
262144			19	24.9	29	26.7	524	1.50	157.3
1048576			37	199.0	49	201.0	720	1.50	833.8
2102500			52	522.0	183	838.0	837	1.50	2208.9
3147076			72	1030.0	338	2090.0	838	1.50	3300.5

Table 5.1: Tandem queueing network. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. Here C_{op} is the operator complexity.

	MCAMG			OC-AGG-f ($\alpha = 2.2$)			OC-AGG-a			IR-AGG-A2		
n	it	C_{op}	time	it	C_{op}	time	it	C_{op}	time	it	C_{op}	time
4096	16	4.85	0.4	45	1.50	0.5	80	1.50	0.8	76	1.50	0.7
16384	18	5.01	2.1	44	1.50	1.8	89	1.50	3.3	72	1.50	2.6
65536	21	5.17	9.8	44	1.50	7.1	108	1.50	13.5	93	1.50	12.2
262144	29	5.12	52.0	43	1.50	25.5	130	1.50	59.1	100	1.50	43.2
1048576	29	5.17	229.9	54	1.50	117.1	142	1.50	248.4	128	1.50	193.5
2102500	30	5.16	510.8	44	1.50	213.6	148	1.50	561.1	156	1.50	452.7
3147076	32	5.19	898.5	44	1.50	328.6	151	1.50	868.5	155	1.50	691.4

Table 5.2: Tandem queueing network. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. Here C_{op} is the operator complexity.

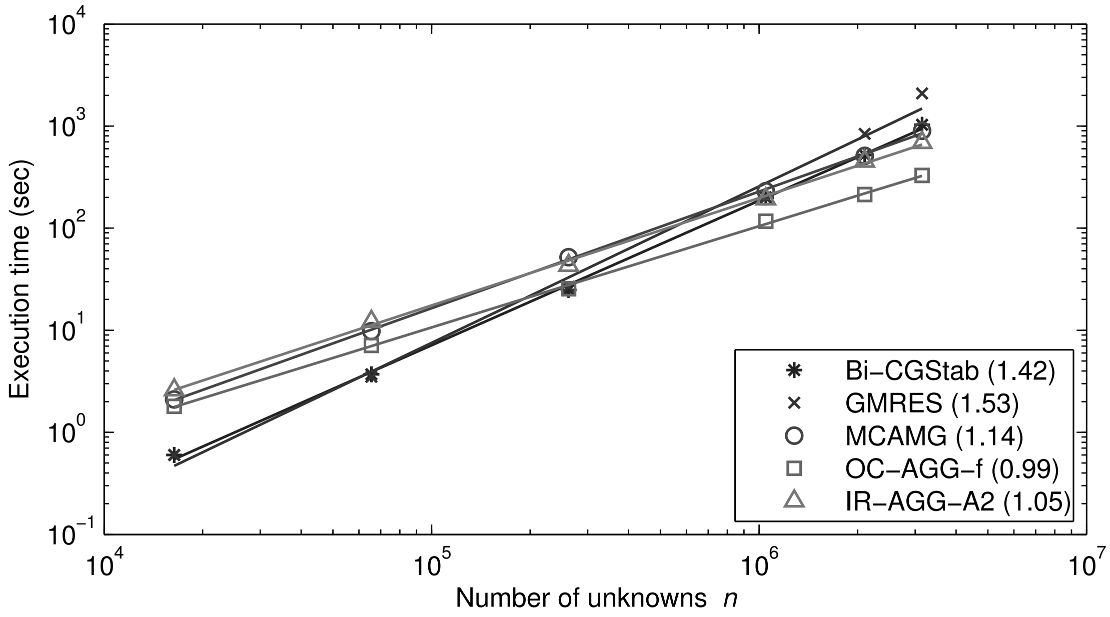


Figure 5.2: Execution time scaling of Bi-CGStab, GMRES, MCAMG, OC-AGG-f, and IR-AGG-A2 for the tandem queueing network. Solid lines are the best-fit lines through the data points (asterisks, crosses, circles, squares, and triangles). Numerical values in the legend are the slopes of the best-fit lines.

5.3.2 Unstructured planar graph

Numerical results for the unstructured directed planar graph are given in Tables 5.3 and 5.4. Preconditioned Bi-CGStab and GMRES are the fastest solvers for the smaller problem sizes; however, the MCAMG method is competitive with Bi-CGStab and is superior to GMRES in terms of execution time for $n = 2102500$, 3147076. Although OC-AGG-f with over-correction parameter $\alpha = 3$ is significantly faster than unaccelerated multilevel aggregation, it is unable to obtain the same textbook multigrid efficiency as for the previous test problem. The OC-AGG-a method demonstrates good speedups and good scalability for smaller problem sizes but struggles for $n \geq 1048576$. Evidently, an over-correction parameter near three is suitable for the larger problem sizes; consequently, restricting the computed over-correction parameter to the interval $[1.1, 2]$ is detrimental to the performance of OC-AGG-a. The numerical tests also revealed that for larger problem sizes it may be beneficial to use a window size larger than two with iterant recombination. In essence, the added numerical cost of solving a larger minimization problem is outweighed by the added savings of performing fewer multilevel iterations. Although Bi-CGStab and GMRES are competitive methods for this test problem in terms of execution time, they do not scale as well as the multilevel methods (Figure 5.3).

	Weighted Jacobi ($\omega = 0.99$)		Bi-CGStab (ILUTP(10^{-3}))		GMRES(50) (ILUTP(10^{-4}))		AGG		
n	it	time	it	time	it	time	it	C_{op}	time
4096	19140	4.4	11	< 0.1	9	0.2	174	1.33	1.6
16384	59950	57.5	21	0.6	15	1.2	310	1.36	9.5
65536	240342	1012.3	37	3.8	25	8.1	469	1.37	56.3
262144			66	21.9	46	51.3	635	1.37	368.5
1048576			145	167.0	110	298.0			
2102500			186	420.0	151	829.0			
3147076			212	744.0	202	1640.0			

Table 5.3: Unstructured directed planar graph. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. Here C_{op} is the operator complexity.

	MCAMG			OC-AGG-f ($\alpha = 3$)			OC-AGG-a			IR-AGG-E3		
n	it	C_{op}	time	it	C_{op}	time	it	C_{op}	time	it	C_{op}	time
4096	21	2.75	0.5	48	1.33	0.6	61	1.33	0.8	55	1.33	0.7
16384	21	2.83	2.0	58	1.35	2.7	91	1.36	4.1	71	1.35	3.2
65536	22	2.84	8.9	56	1.37	11.3	147	1.37	25.1	69	1.37	11.5
262144	27	2.84	44.9	64	1.37	48.9	147	1.37	91.2	89	1.37	52.6
1048576	26	2.83	194.2	103	1.37	269.4	229	1.37	602.7	128	1.37	324.4
2102500	29	2.84	477.7	89	1.37	468.9	175	1.37	1046.9	116	1.37	577.2
3147076	29	2.84	797.1	140	1.37	1036.4	796	1.37	6293.4	145	1.37	1056.1

Table 5.4: Unstructured directed planar graph. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. Here C_{op} is the operator complexity.

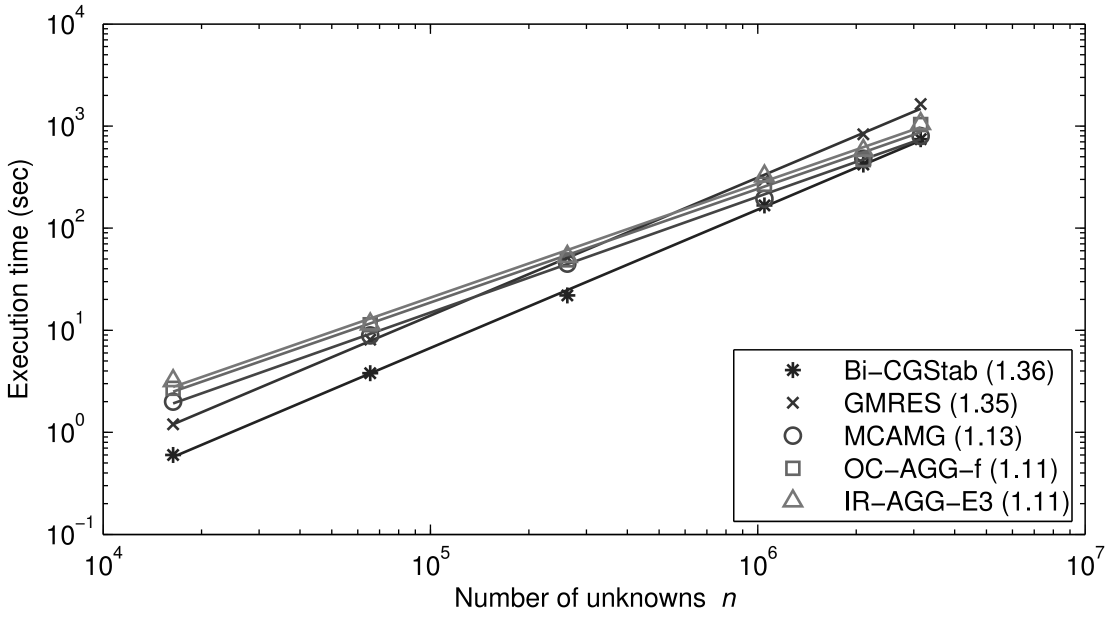


Figure 5.3: Execution time scaling of Bi-CGStab, GMRES, MCAMG, OC-AGG-f, and IR-AGG-E3 for the directed unstructured planar graph. Solid lines are the best-fit lines through the data points (asterisks, crosses, circles, squares, and triangles). Numerical values in the legend are the slopes of the best-fit lines.

5.3.3 Stochastic Petri net

Numerical results for the stochastic Petri net are given in Tables 5.5 and 5.6. Test cases corresponding to $n = 1048061$, 2162281, 3153606 were generated using the initial markings $(145, 0, 0, 0, 0)$, $(185, 0, 0, 0, 0)$, and $(210, 0, 0, 0, 0)$, respectively. In order to obtain smaller operator complexities and faster overall performance of the MCAMG method it was necessary to freeze the transfer operators after nine iterations and use the larger strength threshold $\theta = 0.7$. Unfortunately, despite these steps MCAMG operator complexities remain unacceptably large. Multilevel aggregation with fixed over-correction parameter $\alpha = 1.9$ is the fastest multilevel method for this test problem, with speedups of at least two times over unaccelerated multilevel aggregation. Preconditioned Bi-CGStab is a competitive solver for the smaller problem sizes; however, OC-AGG-f is considerably faster than Bi-CGStab for problem sizes $n = 2162281$, 3153606. In terms of scaling the multilevel methods are superior to Bi-CGStab as shown in Figure 5.4. The generalized minimal

	Weighted Jacobi ($\omega = 0.99$)		Bi-CGStab (ILUTP(0.1))		GMRES(50) (ILUTP(10^{-2}))		AGG		
n	it	time	it	time	it	time	it	C_{op}	time
4324	7394	1.7	24	< 0.1	15	0.2	75	1.64	0.9
16206	16939	16.1	39	0.4	20	2.4	92	1.54	3.5
60116	34625	124.2	51	2.2	27	25.6	79	1.48	10.9
255346			84	16.0	39	404.0	129	1.48	68.7
1048061			190	130.0	82	6010.0	93	1.49	235.7
2162281			248	384.0			115	1.50	583.1
3153606			240	558.0			143	1.50	1029.5

Table 5.5: Stochastic Petri net. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. Here C_{op} is the operator complexity.

	MCAMG ($\theta = 0.7$)			OC-AGG-f ($\alpha = 1.9$)			OC-AGG-a			IR-AGG-A2		
n	it	C_{op}	time	it	C_{op}	time	it	C_{op}	time	it	C_{op}	time
4324	18	2.41	0.4	38	1.87	0.7	36	1.68	0.7	34	1.64	0.6
16206	18	2.49	1.6	39	1.93	2.5	37	1.54	2.4	39	1.54	1.9
60116	23	2.69	8.4	38	1.97	8.6	53	1.48	11.4	37	1.48	6.7
255346	23	2.98	43.7	38	1.82	35.3	45	1.47	43.2	56	1.48	37.2
1048061	21	4.31	268.9	39	1.66	143.4	60	1.49	219.5	53	1.49	154.3
2162281	19	5.48	577.9	38	1.61	299.8	58	1.50	406.6	71	1.50	431.1
3153606	21	5.95	975.0	37	1.59	437.7	68	1.50	684.4	77	1.50	704.2

Table 5.6: Stochastic Petri net. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. Here C_{op} is the operator complexity.

residual method is slow for this test problem because of the long time required to build the ILUTP preconditioner with a small drop tolerance. Transition matrices for the stochastic Petri net test problem have the highest density and the most irregular sparsity pattern of any of the test problems we consider; consequently, significant fill-in may occur during the construction of the ILUTP preconditioner with a small drop tolerance, resulting in a very slow setup phase. We note that in order to obtain convergence with GMRES it was necessary to use a drop tolerance of 0.01 or smaller, whereas Bi-CGStab was able to converge with the relatively large drop tolerance of 0.1.

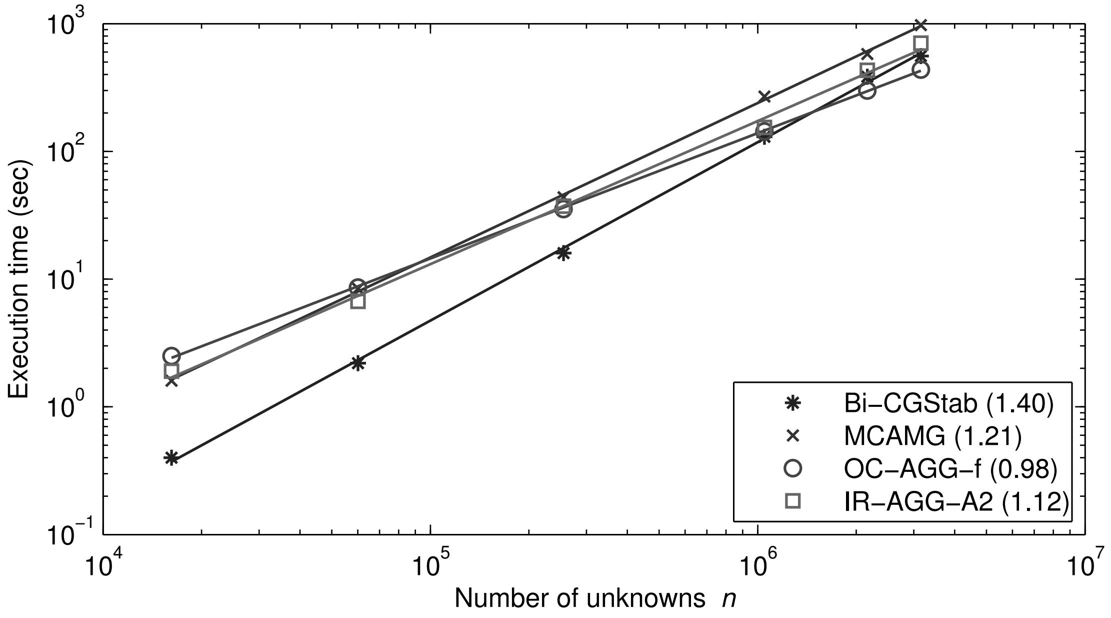


Figure 5.4: Execution time scaling of Bi-CGStab, MCAMG, OC-AGG-f, and IR-AGG-A2 for the stochastic Petri net. Solid lines are the best-fit lines through the data points (asterisks, crosses, circles, and squares). Numerical values in the legend are the slopes of the best-fit lines.

5.3.4 Octagonal mesh

Numerical results for the octagonal mesh are give in Tables 5.7 and 5.8. In this case the MCAMG method demonstrates near-optimal performance and is the fastest multilevel method with speedups of two to four times over unaccelerated multilevel aggregation. Moreover, MCAMG and is competitive with Bi-CGStab and GMRES, and for sufficiently large problem sizes MCAMG is faster than these methods. Although OC-AGG-f is slower than MCAMG for the problem sizes considered, it obtains optimal multilevel scaling as

	Weighted Jacobi ($\omega = 0.99$)		Bi-CGStab (ILUTP(10^{-3}))		GMRES(25) (ILUTP(10^{-4}))		AGG		
n	it	time	it	time	it	time	it	C_{op}	time
4096	10580	1.5	6	< 0.1	6	< 0.1	59	1.97	0.9
16384	42204	26.7	13	0.3	10	0.4	75	1.98	3.5
65536	146946	372.8	25	1.9	16	2.3	119	2.01	17.8
262144			46	14.5	34	18.3	165	2.01	88.5
1048576			92	108.0	83	133.0	219	2.02	529.0
2097152			128	322.0	158	589.0	279	2.02	1285.3
3145728			162	610.0	158	853.0	300	2.02	2018.7

Table 5.7: Octagonal mesh. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. Here C_{op} is the operator complexity.

	MCAMG			OC-AGG-f ($\alpha = 1.3$)			OC-AGG-a			IR-AGG-E3		
n	it	C_{op}	time	it	C_{op}	time	it	C_{op}	time	it	C_{op}	time
4096	20	5.24	0.5	75	1.97	1.0	727	1.97	9.2	48	1.97	0.8
16384	20	5.27	2.4	65	1.98	3.1	647	1.98	25.2	52	1.98	2.7
65536	20	5.08	8.6	66	2.01	12.1	669	2.01	97.3	69	2.01	12.3
262144	20	5.03	32.5	66	2.01	46.2	691	2.01	381.7	86	2.01	54.8
1048576	21	4.91	132.4	72	2.02	192.5	664	2.02	1583.2	107	2.02	270.2
2097152	22	5.03	306.3	85	2.02	446.2	639	2.02	3245.1	125	2.02	667.7
3145728	21	5.00	460.3	81	2.02	618.0	633	2.02	4777.3	131	2.02	992.6

Table 5.8: Octagonal mesh. Iteration counts (it) and execution times in seconds (time) to reduce the residual by a factor of 10^{12}. Here C_{op} is the operator complexity.

shown in Figure 5.5. In contrast to the previous test cases, an over-correction parameter much less than two is optimal for this problem. Consequently, a fixed over-correction parameter near two, which is somewhat standard for multilevel aggregation applied to symmetric positive definite problems, may not always be suitable. The primary reason we have included this test problem is to illustrate that our automatic over-correction approach is not guaranteed to improve the performance of the AGG method. As shown in Table

5.8, the performance of OC-AGG-a is substantially worse than unaccelerated multilevel aggregation. An investigation of this test problem revealed that multilevel aggregation is very sensitive to the choice of the over-correction parameter α. In particular, slow convergence is attributable to values of $\alpha > \alpha_{max}$ that were set to $\alpha_{max} = 2$. Moreover, additional tests of OC-AGG-f with $\alpha = 2$ supports this claim.

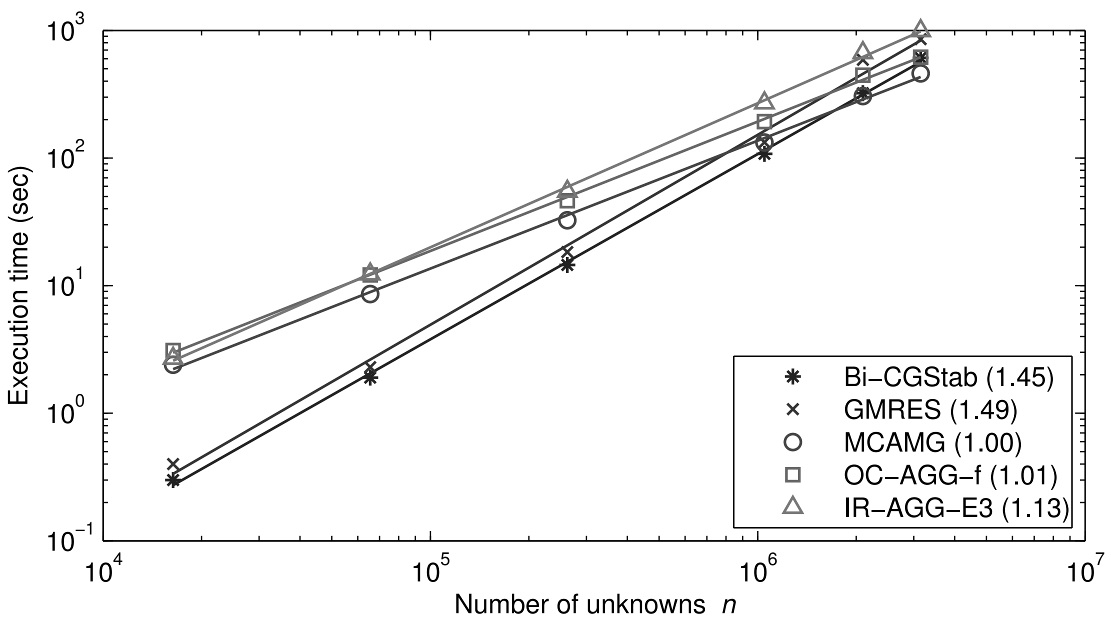


Figure 5.5: Execution time scaling of Bi-CGStab, GMRES, MCAMG, OC-AGG-f, and IR-AGG-E3 for the octagonal mesh. Solid lines are the best-fit lines through the data points (asterisks, crosses, circles, squares, and triangles). Numerical values in the legend are the slopes of the best-fit lines.

5.4 General discussion and conclusions

The main contribution of this chapter was to show that over-correction can significantly improve the convergence of a simple non-overlapping multilevel aggregation method for Markov chains. In particular, we formulated an over-corrected version of the multiplicative coarse-grid correction and developed an automatic over-correction approach that aims to compute an optimal over-correction parameter on each level. Numerical results demonstrated that fixed over-correction, in which the over-correction parameter is determined a priori and is fixed on all levels, can dramatically improve the convergence of simple multilevel aggregation, often resulting in optimal or near-optimal performance with respect to algorithmic scalability. The downside of fixed over-correction is that the over-correction parameter has to be selected through a costly and impractical a priori trial-and-error procedure. It was also shown that automatic over-correction is capable of improving scalability and reducing execution times of simple multilevel aggregation, although not to the extent of fixed over-correction. Although automatic over-correction is clearly computationally more expensive than fixed over-correction, our expectation was that it would reduce iteration counts by as much or even more than fixed over-correction. Unfortunately, in each of the test cases the iteration counts of OC-AGG-a were markedly higher than those of OC-AGG-f, which suggests that more work is needed to improve the automatic over-correction approach. As evidenced by the octagonal mesh problem, robustness of the automatic over-correction approach is an area in which improvements are needed. The numerical results also demonstrated that our multilevel methods tend to scale better than ILU-preconditioned Bi-CGStab and GMRES in terms of execution time. For the smaller problem sizes Bi-CGStab and GMRES were generally the fastest solvers or were very competitive; however, for larger problems our multilevel approach was competitive and often superior to Bi-CGStab and GMRES. Although it is difficult to form general conclusions based on comparisons with a limited number of test problems, it should be noted that the MCAMG and IR-AGG methods consistently showed good scalability and robustness, and were among the fastest multilevel methods for each of the test problems considered. Preliminary tests suggest that combining iterant recombination and over-correction may lead to further improvements over the individual acceleration approaches. In particular, iterant recombination in conjunction with fixed over-correction seems promising, although further testing is required.

A technique to reduce the computational cost of automatic over-correction that seems fruitful is to automatically determine the over-correction parameter on the finest level and use this fixed value on coarser levels. While this approach clearly reduces the per iteration computational cost, preliminary tests suggest it may also be successful at reducing the overall execution time, however, further testing is required. With respect to fixed over-correction, it may be sufficient to choose the over-correction parameter based on an a priori trial-and-error strategy with small problem sizes. Preliminary tests in which a single multilevel aggregation V-cycle is performed for a range of over-correction parameters has worked well for some test problems, but poorly for others. Currently, it is unclear if the value of α obtained from this procedure will be suitable for larger problem sizes. More generally, a rigorous analysis of the over-correction mechanism, or at the very least heuristic arguments in support of our approach, would be of great benefit to guide the formulation of the automatic over-correction minimization problem.

Chapter 6

AMG for Canonical Tensor Decomposition

Recall the definition of the tensor canonical decomposition (CP) from the introduction. Given an Nth-order tensor $\mathfrak{Z}$ and the number of components R, we seek an approximate decomposition of the form

$$\mathfrak{Z} \approx \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \cdots \circ \mathbf{a}_r^{(N)} := [\![\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!],$$

where the factor matrices

$$\mathbf{A}^{(n)} = \left[\mathbf{a}_1^{(n)}, \ldots, \mathbf{a}_R^{(n)}\right] \in \mathbb{R}^{I_n \times R} \quad \text{for} \quad n = 1, \ldots, N. \tag{6.1}$$

The problem of computing CP with R components that best approximates an arbitrary Nth-order tensor $\mathfrak{Z}$ is formulated as a nonlinear least squares optimization problem:

$$\text{minimize} \quad f(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}) := \frac{1}{2}\left\|\mathfrak{Z} - [\![\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!]\right\|^2. \tag{6.2}$$

We note that in this chapter all norms $\|\cdot\|$ correspond to the Frobenius norm (see §2.1). A general approach to solving this optimization problem is to find solutions of the *first-order* *optimality equations*, that is, to find a set of nontrivial factor matrices that zero out the gradient of f [1, 79].

In this chapter we describe an adaptive algebraic multigrid method for computing (local) minimizers of f by solving the optimality equations. Our multigrid method consists of two multilevel phases: a multiplicative correction scheme as the setup phase and an additive correction scheme as the solve phase. We note that this combination of multiplicative and additive methods is similar to the hybrid method discussed in Chapter 3 for Markov chains. In the setup phase a multiplicative correction scheme is used in conjunction with bootstrap algebraic multigrid (BAMG) interpolation [20, 72] not only to build the necessary transfer operators and coarse-level tensors but also to compute initial approximations of the factor matrices. In this phase the alternating least squares (ALS) method forms the relaxation scheme on all levels. The setup phase is adaptive in the sense that the transfer operators are continually improved using the most recent approximation to the solution factor matrices. In order for the exact solution to be a fixed point of the multiplicative correction scheme, it must lie exactly in the range of interpolation at convergence. However, because each interpolation operator attempts to fit multiple factors (in a least squares sense), this condition can be met only approximately. Therefore, after a few setup cycles the transfer operators and coarse-level tensors are frozen, and additive correction cycles are used in the solve phase, which can still converge when the exact solution lies only approximately in the range of interpolation. The combination of a multiplicative setup scheme and BAMG has already been considered in [83], where it formed the basis of an efficient eigensolver for multiclass spectral clustering problems. A similar approach was also proposed in [21, 72]. In the solve phase we use the full approximation scheme [18, 23] to efficiently obtain an accurate solution (see §2.6.5). Our multigrid framework is closely related to recent work on an adaptive AMG solver for extremal singular triplets and eigenpairs of matrices [47], and to a lesser degree to multigrid methods for Markov chains [15, 53, 118]. We note that while our proposed algorithm is the first such multigrid method for computing the CP decomposition of a tensor, the idea of applying multilevel methods to problems in multilinear algebra has already been discussed and analyzed [4, 16, 77]. In the context of linear systems that arise from the discretization of high-dimensional PDE problems, it is often possible to overcome the *curse of dimensionality* by exploiting the tensor structure of the linear system when constructing the components of the multigrid method. Although our method applies multigrid techniques to tensors, we do not expect its performance to improve as the dimensionality increases, primarily because our method is currently formulated for general tensors and does not exploit any tensor product structure or hierarchical tensor structure that the tensors may possess. In particular, we show in §6.4.1 that a single solution cycle has complexity $\mathcal{O}(NPR)$, where $P = s^N$ for an Nth-order tensor with each mode size equal to s.

We expect our method to work well for tensors with properties that make a multilevel approach beneficial, but less so for generic tensors that lack these properties. Just as in the case of multigrid for matrix systems derived from PDE discretization, our multilevel approach can lead to significant speedup when error components that are damped only weakly by the fine-level process can be represented and damped efficiently on coarser levels. We expect this to be the case for the decomposition of certain higher-order tensors that arise in the context of PDE discretization on high-dimensional regular lattices [75, 76, 97, 98], and we will illustrate the potential benefits of the proposed multigrid method for these types of problems. To illustrate the potential applicability of our approach to broader classes of tensors, we also present some numerical tests for a standard non-PDE tensor decomposition problem. It should also be noted that since a single interpolation operator is associated with an entire factor matrix, and since each interpolation operator can only be expected to represent a small number of factors in a sufficiently accurate way, especially if the desired factors have little in common, the multigrid acceleration proposed here will only be effective for low-rank decompositions with small R (e.g., up to 5 or 6). These restrictions are entirely analogous to the case of the adaptive multigrid method for computing SVD triplets of a matrix [47].

The remainder of this chapter is organized as follows. We begin by stating the first-order optimality equations for CP and describing the alternating least squares method in §6.1. Section 6.2 describes the multilevel setup phase, and §6.3 describes the multilevel solve phase. Implementation details and numerical results are presented in §6.4, followed by concluding remarks in §6.5.

6.1 CP first-order optimality equations and ALS

The first-order optimality equations for CP are obtained by setting the gradient of the functional in (6.2) equal to zero. Following the derivation of ∇f given in [1], for each mode $n \in \{1, \ldots, N\}$ the derivative of f with respect to $\mathbf{A}^{(n)}$ can be written as an $I_n \times R$ matrix

$$\mathbf{G}^{(n)} = -\mathbf{Z}_{(n)}\mathbf{\Phi}^{(n)} + \mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)}, \tag{6.3}$$

where

$$\mathbf{\Phi}^{(n)} = \mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)} \tag{6.4}$$

and

$$\mathbf{\Gamma}^{(n)} = \mathbf{\Upsilon}^{(1)} * \cdots * \mathbf{\Upsilon}^{(n-1)} * \mathbf{\Upsilon}^{(n+1)} * \cdots * \mathbf{\Upsilon}^{(N)}, \tag{6.5}$$

with $\mathbf{\Upsilon}^{(n)} = {\mathbf{A}^{(n)}}^\top \mathbf{A}^{(n)} \in \mathbb{R}^{R\times R}$ for $n = 1, \ldots, N$. Recall that $\odot$ is the Khatri–Rao product, which is equivalent to the columnwise Kronecker product (see §2.4). The first-order optimality equations are then given by

$$\mathbf{G}^{(n)} = \mathbf{0} \quad \text{for} \quad n = 1, \ldots, N. \tag{6.6}$$

We note that the $R \times R$ matrix $\mathbf{\Gamma}^{(n)}$ is symmetric positive semidefinite because it is the Hadamard (elementwise) product $*$ of symmetric positive semidefinite matrices [113]. Additionally, if each factor matrix $\mathbf{A}^{(n)}$ has full rank then $\mathbf{\Gamma}^{(n)}$ is symmetric positive definite.

One iteration of ALS for the CP decomposition is equivalent to one iteration of block nonlinear Gauss–Seidel applied to the optimality equations (6.6). Iterating through the modes sequentially, at the nth step the factor matrices are fixed for all modes except n, and the resulting linear least squares problem is solved for $\mathbf{A}^{(n)}$. The linear least squares problem is solved by updating $\mathbf{\Gamma}^{(n)}$ and $\mathbf{\Phi}^{(n)}$ and setting $\mathbf{A}^{(n)} \leftarrow \mathbf{Z}_{(n)}\mathbf{\Phi}^{(n)}(\mathbf{\Gamma}^{(n)})^\dagger$, where $(\mathbf{\Gamma}^{(n)})^\dagger$ is the Moore–Penrose pseudoinverse of $\mathbf{\Gamma}^{(n)}$. We note that the order in which the factor matrices are updated during an ALS iteration can be any permutation of the indices $1, \dots, N$. Because of the scaling indeterminacy inherent to CP, during ALS some factors may tend to infinity while others may compensate by tending to zero, such that the rank-one components remain bounded. This behavior can be avoided by using a normalization strategy. After each ALS iteration the factors of the rth component are normalized according to

$$\mathbf{a}_r^{(n)} \mapsto \lambda_r \left(\frac{\mathbf{a}_r^{(n)}}{\|\mathbf{a}_r^{(n)}\|} \right) \quad \text{for} \quad n = 1, \dots, N, \quad \lambda_r = \left(\|\mathbf{a}_r^{(1)}\| \dots \|\mathbf{a}_r^{(N)}\| \right)^{1/N} \tag{6.7}$$

for $r = 1, \dots, R$. This normalization equilibrates the norms of the factors of each component, that is,

$$\|\mathbf{a}_r^{(1)}\| = \|\mathbf{a}_r^{(2)}\| = \dots = \|\mathbf{a}_r^{(N)}\| \quad \text{for} \quad r = 1, \dots, R.$$

Because of the permutation indeterminacy inherent to CP, upon completion of ALS the rank-one terms are sorted in decreasing order of the normalization factors λ_r. The ALS method described here is used as the relaxation method and coarsest-level solver in the setup phase. A local convergence analysis of ALS for the tensor canonical decomposition is discussed in [120].

6.2 Multiplicative setup phase

This section describes the multilevel hierarchy constructed in the setup phase of our solver. Two-level notation is used to describe the interaction of two levels at a time. Coarse-level quantities are denoted by a subscript "c", except in cases where a superscript "c" improves readability. Fine-level quantities and transfer operators have neither subscripts nor superscripts. We note that our setup phase is similar to the setup phase of [47] for computing SVD triplets of a matrix in that interpolation matrices are constructed via bootstrap AMG and a separate interpolation matrix is defined for each mode. Furthermore, the formulation of the coarse-level equations and the two-level multiplicative coarse-grid correction scheme is similar to the formulation of the MCAMG method.

6.2.1 Derivation of coarse-level equations

The fine-level equations correspond to the first-order optimality equations (see §6.1) given by

$$\mathbf{Z}_{(n)}\Big(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}\Big) = \mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)} \tag{6.8}$$

for $n = 1, \ldots, N$. Suppose there exist N full-rank operators $\mathbf{P}^{(n)} \in \mathbb{R}^{I_n \times I_{n,c}}$ with $1 < I_{n,c} < I_n$, such that $\mathbf{A}^{(n)}$ lies approximately in the range of $\mathbf{P}^{(n)}$, that is, $\mathbf{A}^{(n)} \approx \mathbf{P}^{(n)}\mathbf{A}_c^{(n)}$ for some coarse-level variable $\mathbf{A}_c^{(n)} \in \mathbb{R}^{I_{n,c} \times R}$ and for each n. (Since each factor matrix has R columns it is unlikely that equality can be achieved.) Then a solution of (6.8) can be approximated by solving a coarse-level problem given by

$$\begin{aligned}&\mathbf{P}^{(n)\top}\mathbf{Z}_{(n)}\Big(\mathbf{P}^{(N)}\mathbf{A}_c^{(N)} \odot \cdots \odot \mathbf{P}^{(n+1)}\mathbf{A}_c^{(n+1)} \odot \mathbf{P}^{(n-1)}\mathbf{A}_c^{(n-1)} \odot \cdots \odot \mathbf{P}^{(1)}\mathbf{A}_c^{(1)}\Big)\\ &= \Big(\mathbf{P}^{(n)\top}\mathbf{P}^{(n)}\Big)\mathbf{A}_c^{(n)}\mathbf{\Gamma}_c^{(n)} \quad \text{for} \quad n = 1, \ldots, N,\end{aligned} \tag{6.9}$$

followed by interpolation. Here, $\mathbf{\Gamma}_c^{(n)}$ is defined as in (6.5) with

$$\mathbf{\Upsilon}_c^{(n)} = \mathbf{A}_c^{(n)\top}\Big(\mathbf{P}^{(n)\top}\mathbf{P}^{(n)}\Big)\mathbf{A}_c^{(n)} \quad \text{for} \quad n = 1, \ldots, N.$$

Letting $\mathbf{B}^{(n)} = \mathbf{P}^{(n)\top}\mathbf{P}^{(n)}$ for each mode n, and using properties (2.18) and (2.19), the coarse-level problem (6.9) can be written as

$$\mathbf{Z}^c_{(n)}\Big(\mathbf{A}_c^{(N)} \odot \cdots \odot \mathbf{A}_c^{(n+1)} \odot \mathbf{A}_c^{(n-1)} \odot \cdots \odot \mathbf{A}_c^{(1)}\Big) = \mathbf{B}^{(n)}\mathbf{A}_c^{(n)}\mathbf{\Gamma}_c^{(n)}, \tag{6.10}$$

where the coarse-level tensor is given by the product

$$\mathfrak{Z}^c = \mathfrak{Z} \times_1 \mathbf{P}^{(1)\top} \times_2 \mathbf{P}^{(2)\top} \cdots \times_N \mathbf{P}^{(N)\top}. \tag{6.11}$$

Note that (6.11) is essentially a higher-dimensional analogue of the Galerkin coarse-level operator that is commonly used in algebraic multigrid for the matrix case. By the full-rank assumption on the interpolation operators it follows that $\mathbf{B}^{(n)}$ is SPD; hence we can compute its Cholesky factor $\mathbf{L}^{(n)}$, which is an $I_{n,c} \times I_{n,c}$ nonsingular lower triangular matrix. The Cholesky factors are used to transform (6.10), whereby one obtains an equivalent set of equations that correspond to the first-order optimality equations of a coarse-level CP optimization problem. Defining the transformed coarse-level factor matrices by $\hat{\mathbf{A}}_c^{(n)} = \mathbf{L}^{(n)\top} \mathbf{A}_c^{(n)}$ for $n = 1, \ldots, N$, and again appealing to properties (2.18) and (2.19), it follows that (6.10) can be written as

$$\hat{\mathbf{Z}}^c_{(n)} \left(\hat{\mathbf{A}}_c^{(N)} \odot \cdots \odot \hat{\mathbf{A}}_c^{(n+1)} \odot \hat{\mathbf{A}}_c^{(n-1)} \odot \cdots \odot \hat{\mathbf{A}}_c^{(1)} \right) = \hat{\mathbf{A}}_c^{(n)} \hat{\mathbf{\Gamma}}_c^{(n)},$$

where the transformed coarse-level tensor is given by

$$\hat{\boldsymbol{\mathcal{Z}}}^c = \boldsymbol{\mathcal{Z}} \times_1 \hat{\mathbf{P}}^{(1)\top} \times_2 \hat{\mathbf{P}}^{(2)\top} \cdots \times_N \hat{\mathbf{P}}^{(N)\top}, \tag{6.12}$$

with $\hat{\mathbf{P}}^{(n)} = \mathbf{P}^{(n)} \mathbf{L}^{(n)-\top}$ for $n = 1, \ldots, N$. Note that $\hat{\mathbf{\Gamma}}_c^{(n)} = \mathbf{\Gamma}_c^{(n)}$ for all modes n. Hence, the coarse-level equations are equivalent to the gradient equations of the following coarse-level functional:

$$\hat{f}_c(\hat{\mathbf{A}}_c^{(1)}, \ldots, \hat{\mathbf{A}}_c^{(N)}) \coloneqq \frac{1}{2} \left\| \hat{\boldsymbol{\mathcal{Z}}}^c - [\![\hat{\mathbf{A}}_c^{(1)}, \ldots, \hat{\mathbf{A}}_c^{(N)}]\!] \right\|^2. \tag{6.13}$$

Therefore, the coarse-level equations can be solved by applying ALS to minimize $\hat{f}_c$. An initial guess for the mode-n coarse-level factor matrix is obtained by applying a restriction operator $\hat{\mathbf{R}}^{(n)}$, defined as the transpose of $\hat{\mathbf{P}}^{(n)}$, to the current fine-level approximation of $\mathbf{A}^{(n)}$. We note that since $\hat{\mathbf{R}}^{(n)} \hat{\mathbf{P}}^{(n)}$ is equal to the coarse-level identity, it follows that $\hat{\mathbf{R}}^{(n)} \mathbf{u} = \hat{\mathbf{R}}^{(n)} \hat{\mathbf{P}}^{(n)} \hat{\mathbf{u}}_c = \hat{\mathbf{u}}_c$ for any vector $\mathbf{u}$ in the range of $\mathbf{P}^{(n)}$. Moreover, $\hat{\mathbf{P}}^{(n)} \hat{\mathbf{R}}^{(n)} \mathbf{u} = \mathbf{u}$ for any vector $\mathbf{u}$ in the range of $\mathbf{P}^{(n)}$. After solving the coarse-level equations, the coarse-grid corrected fine-level approximations are obtained via prolongation:

$$\mathbf{A}_{\text{CGC}}^{(n)} = \hat{\mathbf{P}}^{(n)} \hat{\mathbf{A}}_c^{(n)} \quad \text{for} \quad n = 1, \ldots, N. \tag{6.14}$$

Now suppose that $\mathbf{P}^{(n)}$ contains $\mathbf{A}^{(n)}$ exactly in its range, and hence $\hat{\mathbf{R}}^{(n)}\mathbf{A}^{(n)} = \hat{\mathbf{A}}_c^{(n)}$. Further suppose that $\hat{\mathbf{A}}_c^{(n)}$ is a fixed point of the coarse-level solver. Then

$$\mathbf{A}_{\text{CGC}}^{(n)} = \hat{\mathbf{P}}^{(n)}\hat{\mathbf{A}}_c^{(n)} = (\hat{\mathbf{P}}^{(n)}\hat{\mathbf{R}}^{(n)})\mathbf{A}^{(n)} = \mathbf{A}^{(n)} \quad \text{for} \quad n = 1, \ldots, N.$$

However, because these assumptions are satisfied only approximately, we expect (6.14) to yield an improved but not exact approximation to the fine-level solution. In particular, because the approximation properties of the interpolation operators deteriorate as the number of components increases, we expect our method to perform well for a relatively small number of components R.

6.2.2 Bootstrap AMG V-cycles

The multiplicative setup phase uses the bootstrap AMG approach described in [20, 21, 26] to find initial approximations of the desired factor matrices, and to adaptively determine the interpolation operators that approximately fit the factor matrices. In the context of a linear system $\mathbf{A}\mathbf{x} = \mathbf{f}$, Bootstrap AMG is a general interpolation scheme that fits best (in a least-squares sense) a set of relaxed error vectors, referred to as *test vectors*, each of which is obtained by relaxing the homogeneous system of equations from a random initial guess. The BAMG process for computing the test vectors proceeds by applying relaxation to the homogeneous system

$$\mathbf{A}_\ell \mathbf{x}_\ell = \mathbf{0}_\ell$$

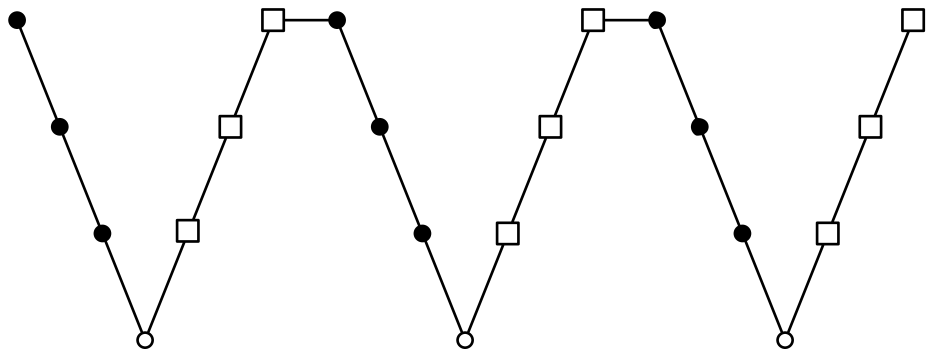
on each level $\ell = 0, \ldots, L-1$. Assuming that a priori knowledge of the algebraically smooth error is not available, the test vectors are initialized randomly on the finest level, and are obtained on coarser levels by restricting the test vectors computed on the previous fine level. Interpolation is constructed through a least-squares fitting of the test vectors, and the coarse-level system operators are computed via the variational Galerkin definition. Once an initial multigrid hierarchy has been computed, the current set of test vectors is further enhanced on all levels using the existing multigrid structure.

We now describe the initial BAMG V-cycle for the multilevel setup phase. On the finest level we start with n_t *test blocks* (a generalization of test vectors), where each test block is a collection of N randomly generated *test factor matrices* (TFMs) $\mathbf{A}_t^{(1)}, \ldots, \mathbf{A}_t^{(N)}$. It is necessary to use test blocks instead of adding more columns to the factor matrices because the rank-one components of the best rank-R CP tensor approximation must be found simultaneously [79]; contrary to the best rank-R matrix approximation, the best rank-R CP approximation cannot be obtained by truncating the best rank-Q approximation with $Q > R$. We also start with a collection of N randomly generated *boot factor matrices* (BFMs) $\mathbf{A}_b^{(1)}, \ldots, \mathbf{A}_b^{(N)}$, which serve as our initial guess to the desired factor matrices. We note that the subscripts "t" and "b" serve only to distinguish between the test and boot factors.

In the downward sweep of the first BAMG V-cycle, the BFMs and the test blocks are relaxed (each test block is relaxed individually) by a few iterations of the ALS algorithm described in §6.1. The modes are coarsened and the interpolation operators $\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(n)}$ are constructed. The nth interpolation operator $\mathbf{P}^{(n)}$ fits the factors in the nth TFMs across all test blocks in a least squares sense such that these factors lie approximately in the range of $\mathbf{P}^{(n)}$. The coarse-level tensor $\hat{\boldsymbol{\mathfrak{Z}}}^c$ is constructed, and the TFMs and BFMs are restricted to the coarse level. This process is then repeated recursively on each level until the coarsest level is reached, from which point on we relax only on the BFMs.

In the upward sweep of the first cycle, starting from the coarsest level, the BFMs are recursively interpolated up to the next finer level, which gives the coarse-grid corrected approximation on that level. The coarse-grid corrected BFMs are then relaxed by a few iterations of ALS. This process continues until the coarse-grid correction on the finest level has been relaxed by ALS.

The initial BAMG V-cycle can be followed by several additional BAMG V-cycles (see Figure 6.1). These cycles are the same as the initial cycle except for one key difference. In the downward sweep the nth interpolation operator $\mathbf{P}^{(n)}$ fits the factors in the nth TFMs across all test blocks as well as the factors in the nth BFM. Since the BFMs serve as the initial approximation of the solution for the additive phase of the algorithm, they must be well represented by interpolation if the additive solve phase is to converge.



● Relax on the TFMs and BFMs

○ Solve for BFMs via ALS

□ Relax on BFMs

Figure 6.1: Illustration of BAMG V-cycles for the multiplicative setup phase.

6.2.3 Interpolation sparsity structure: Coarsening

Construction of the interpolation operators proceeds in two phases. In the first phase the sparsity structure of $\mathbf{P}^{(n)}$ is determined by partitioning the set of fine-level indices $\Omega_n = \{1, \ldots, I_n\}$ into a set of C-points, $\mathcal{C}_n$, with cardinality $1 < I_{n,c} < I_n$, and a set of F-points, $\mathcal{F}_n$. For each point $i \in \mathcal{F}_n$ we define a set of coarse interpolatory points $\mathcal{C}_n^i$, which contains coarse points that i interpolates from. For convenience we assume that the points in $\mathcal{C}_n^i$ are labeled by their coarse-level indices. Furthermore, for any fine-level point $i \in \mathcal{C}_n$ we let $\alpha(i)$ denote its coarse-level index. The interpolation operator $\mathbf{P}^{(n)}$ is defined by

$$p_{ij}^{(n)} = \begin{cases} w_{ij}^{(n)} & \text{if } i \in \mathcal{F}_n \text{ and } j \in \mathcal{C}_n^i, \\ 1 & \text{if } i \in \mathcal{C}_n \text{ and } j = \alpha(i), \\ 0 & \text{otherwise,} \end{cases}$$

where the $w_{ij}^{(n)}$s are the interpolation weights for mode n. The interpolation weights are determined by the least squares process described in §6.2.4. The coarse degrees of freedom are obtained through standard geometric coarsening of each mode, whereby $\mathcal{C}_n$ consists of the odd-numbered points in Ω_n and $\mathcal{F}_n$ consists of the even-numbered points (hence $\alpha(i) = (i+1)/2$). For each $i \in \mathcal{F}_n$ we define $\mathcal{C}_n^i = \{\alpha(i-1), \alpha(i+1)\}$ (coarse-level labels) except possibly at the right endpoint. This coarsening works well when the modes have approximately the same size; however, when some of the mode sizes vary widely a more aggressive coarsening of the larger modes may be appropriate. In §6.2.5 we describe a straightforward approach to coarsening tensors with varying mode sizes. While the simple coarsening procedure discussed here is suitable for the structured test problems considered in §6.4 (PDE problems on high-dimensional regular lattices), more general coarsening algorithms for other types of tensors are desirable. The development of such algorithms is a topic of future research.

6.2.4 Least squares determination of interpolation weights

Suppose that mode n has been coarsened and that $\mathcal{C}_n$ and $\mathcal{F}_n$ are given. Further suppose that the factors in the nth TFMs across all test blocks are stored as the columns of the $I_n \times Rn_t$ matrix $\mathbf{U}_t$, and let $\mathbf{U}_b = \mathbf{A}_b^{(n)}$ for the BFMs. Following [20, 21, 26, 47] we use a least squares approach to determine the interpolation weights in the rows of $\mathbf{P}^{(n)}$ that correspond to points in $\mathcal{F}_n$. The weights are chosen such that the vectors in $\mathbf{U}_t$ and $\mathbf{U}_b$ (except in the first cycle) lie approximately in the range of $\mathbf{P}^{(n)}$. Let the columns of $\mathbf{U}_f = [\mathbf{U}_t \mid \mathbf{U}_b]$ hold the $n_f = R(n_t+1)$ vectors to be fitted. Let $\mathbf{u}_k$ be the kth column of $\mathbf{U}_f$, and let u_{ik} be the value of $\mathbf{u}_k$ at the fine-level point i. Let $\mathbf{u}_{k,c}$ be the coarse-level version of $\mathbf{u}_k$ obtained by injection, and let $(\mathbf{u}_{k,c})_j$ be its value at the coarse-level point j. The interpolation weights of each row that corresponds to a point in $\mathcal{F}_n$ may now be determined consecutively by independent least squares fits. For each point $i \in \mathcal{F}_n$ with coarse interpolatory set $\mathcal{C}_n^i$, the following least-squares problem is solved for the unknown interpolation weights $w_{ij}^{(n)}$:

$$u_{ik} = \sum_{j \in \mathcal{C}_n^i} w_{ij}^{(n)} (\mathbf{u}_{k,c})_j \quad \text{for} \quad k = 1, \ldots, n_f. \tag{6.15}$$

We make (6.15) overdetermined by choosing $n_t > M_s/R$, where M_s is the maximum interpolation stencil size for any i on any level, that is, $|\mathcal{C}_n^i| \leq M_s$. Owing to the standard geometric coarsening of each mode, $M_s = 2$, so it is sufficient to use $n_t = 2$ for any number of components $R > 1$. For a rank-one decomposition we must take $n_t \geq 3$.

In practice (6.15) is formulated as a *weighted* least squares problem, where the weights should bias the fit toward the boot factors. For a fixed n, the weights for the mode-n factor vectors are given by

$$\mu_r = \frac{\|\mathbf{A}^{(n)}\|^2}{\|\mathbf{G}^{(n)}\|^2} \quad \text{for} \quad r = 1, \ldots, R, \tag{6.16}$$

where $\mathbf{G}^{(n)}$ is the gradient in (6.3). Weights are computed for each test block as well as for the BFMs. The weights for all test blocks are stored in the vector $\boldsymbol{\mu}_t$ of length Rn_t, and the weights for the boot factors are stored in the vector $\boldsymbol{\mu}_b$ of length R. The full vector of weights $\boldsymbol{\mu} \in \mathbb{R}^{n_f}$ is obtained by "stacking" $\boldsymbol{\mu}_t$ on top of $\boldsymbol{\mu}_b$. Equation (6.16) stems from the observation that $\mathbf{G}^{(n)}$ is a residual for the nth factor matrix. Therefore, since the BFMs should converge much faster than the TFMs, the gradient norm for the BFMs should be much smaller, and hence their weights should be larger. We note that weights corresponding to a single factor matrix are chosen identical in (6.16) since we do not want preferential treatment given to different factor vectors, but rather to entire factor matrices.

Defining the $n_f \times n_f$ diagonal weight matrix $\mathbf{M} = \text{diag}(\boldsymbol{\mu})$, the $|\mathcal{C}_n^i| \times n_f$ coarse matrix $\mathbf{U}_c = [\mathbf{u}_{1,c}, \ldots, \mathbf{u}_{n_f,c}]$, the row vector $\mathbf{w}_i$ of interpolation weights, and the row vector $\mathbf{u}_i$ as the ith row of $\mathbf{U}_f$, it follows that $\mathbf{w}_i$ is chosen to minimize the functional

$$F(\mathbf{w}_i) = \left\|\mathbf{u}_i\mathbf{M}^{1/2} - \mathbf{w}_i\mathbf{U}_c\mathbf{M}^{1/2}\right\|_2^2. \tag{6.17}$$

Setting the gradient of F equal to zero, the critical points of F satisfy

$$\mathbf{w}_i\mathbf{U}_c\mathbf{M}\mathbf{U}_c^\top = \mathbf{u}_i\mathbf{M}\mathbf{U}_c^\top. \tag{6.18}$$

This linear system has a unique solution if $\text{rank}(\mathbf{U}_c\mathbf{M}^{1/2}) = |\mathcal{C}_n^i|$, or equivalently, if $\mathbf{U}_c$ has full rank and $\mathbf{M}$ is nonsingular. By making the least squares problems overdetermined, it is likely that $\mathbf{U}_c$ will have full rank. In our implementation the local least squares problems (6.18) are solved via QR factorization. That is, we compute the *reduced* QR factorization $\mathbf{U}_c\mathbf{M}\mathbf{U}_c^\top = \tilde{\mathbf{Q}}\tilde{\mathbf{R}}$, and solve the upper-triangular system

$$\tilde{\mathbf{R}}\mathbf{w}_i^\top = \tilde{\mathbf{Q}}^\top(\mathbf{U}_c\mathbf{M}\mathbf{u}_i^\top).$$

We note that solving the local least squares problems (6.18) via QR factorization is mathematically equivalent to right multiplying $\mathbf{u}_i\mathbf{M}\mathbf{U}_c^\top$ by the Moore–Penrose pseudoinverse of $\mathbf{U}_c\mathbf{M}\mathbf{U}_c^\top$, and provides better numerical stability than solving the normal equations directly.

6.2.5 CP-AMG-mult algorithm

A pseudocode description for a multiplicative setup phase V-cycle with ALS as the relaxation scheme and coarsest-level solver is given by Algorithm 6.1. The CP-AMG-mult algorithm recursively coarsens each mode until it reaches some predefined coarsest level. Since the size of each mode $I_1, \ldots, I_N$ may differ and since the rate of coarsening is the same for each mode, it is possible that some modes may reach their coarsest level sooner than others. Therefore, for each mode n we define a threshold $I_{n,coarsest}$ to be the maximum size of that mode's coarsest level, and we continue to coarsen that mode until $I_n \leq I_{n,coarsest}$. Let the modes that still require further coarsening be indexed by the set $\mathcal{I}_c = \{n : I_n > I_{n,coarsest}\}$, and let $\mathcal{I}_c'$ denote its complement. Then for each $n \in \mathcal{I}_c'$, and at any given level, it follows that $\hat{\mathbf{P}}^{(n)} = \mathbf{I}^{(n)}$, where $\mathbf{I}^{(n)}$ is the $I_n \times I_n$ identity matrix. Setting $\hat{\mathbf{P}}^{(n)}$ equal to the identity for all $n \in \mathcal{I}_c'$ has the following implications. The coarse-level tensor is obtained by taking the product in (6.12) over the modes in $\mathcal{I}_c$, instead of for all $n = 1, \ldots, N$. The coarse-level approximations of the BFMs are given by

$$\tilde{\mathbf{A}}_c^{(n)} = \begin{cases} \hat{\mathbf{R}}^{(n)}\mathbf{A}^{(n)} & \text{if } n \in \mathcal{I}_c, \\ \mathbf{A}^{(n)} & \text{if } n \in \mathcal{I}_c' \end{cases} \quad \text{for} \quad n = 1, \ldots, N. \tag{6.19}$$

Similarly, the coarse-level approximations of the TFMs in each test block are computed by restricting only those factor matrices indexed by $\mathcal{I}_c$. Additionally, the coarse-grid corrected BFMs are given by

$$\mathbf{A}^{(n)}_{\text{CGC}} = \begin{cases} \hat{\mathbf{P}}^{(n)}\hat{\mathbf{A}}^{(n)}_c & \text{if } n \in \mathcal{I}_c, \\ \hat{\mathbf{A}}^{(n)}_c & \text{if } n \in \mathcal{I}'_c \end{cases} \quad \text{for} \quad n = 1, \dots, N. \tag{6.20}$$

Algorithm 6.1: V-cycle for setup phase of CP decomposition (CP-AMG-mult)

```
    Input: tensor $\mathcal{Z}$, BFMs $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}$, TFMs
    Output: updated BFMs $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}$, updated TFMs
 1. Compute the set $\mathcal{I}_c = \{n : I_n > I_{n,coarsest}\}$
    if $\mathcal{I}_c \neq \emptyset$ then
 2.     Apply $\nu_1$ relaxations to TFMs in each test block and to $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}$
        for $n \in \mathcal{I}_c$ do
 3.         Build the interpolation operator $\mathbf{P}^{(n)}$ (on first cycle only use TFMs)
 4.         Let $\mathbf{B}^{(n)} \leftarrow \mathbf{P}^{(n)\top}\mathbf{P}^{(n)}$ and compute the Cholesky factor $\mathbf{L}^{(n)}$ of $\mathbf{B}^{(n)}$
 5.         Let $\hat{\mathbf{P}}^{(n)} \leftarrow \mathbf{P}^{(n)}\mathbf{L}^{(n)-\top}$ and $\hat{\mathbf{R}}^{(n)} \leftarrow \hat{\mathbf{P}}^{(n)\top}$
        end
 6.     Compute the coarse BFMs and coarse TFMs according to (6.19)
 7.     Compute the coarse-level tensor $\hat{\mathcal{Z}}^c \leftarrow \mathcal{Z} \times_{n\in\mathcal{I}_c} \hat{\mathbf{R}}^{(n)}$
 8.     Recursive solve:
            $\{\hat{\mathbf{A}}^{(1)}_c, \dots, \hat{\mathbf{A}}^{(N)}_c\} \leftarrow$ CP-AMG-mult($\hat{\mathcal{Z}}_c$, $\tilde{\mathbf{A}}^{(1)}_c, \dots, \tilde{\mathbf{A}}^{(N)}_c$, coarse TFMs)
 9.     Compute the CGC $\mathbf{A}^{(n)}$ for $n = 1, \dots, N$ according to (6.20)
10.     Apply $\nu_2$ relaxations to $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}$
    else
11.     Apply $\nu_c$ relaxations to $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}$
    end
```

The size of the coarsest level plays an important role in multigrid performance. If the coarsest level is too large, then not enough work is done on the coarser levels and convergence will be slow. Conversely, choosing too small a coarsest level may negatively impact convergence, or in some cases may even cause divergence (as in [47, 72, 83]). In practice we have found that choosing $I_{n,coarsest} \geq R$ for all n works well.

6.3 Full approximation scheme additive solve phase

In this section we describe how the full approximation scheme (see §2.6.5) can be used to obtain an additive correction method for the CP decomposition. Two-level notation is used to describe the interaction of two levels at a time with coarse-level quantities denoted by a subscript "c".

6.3.1 Coarse-level equations

Recall the finest-level equations (6.8), and suppose we define nonlinear operators

$$\mathbf{H}^{(1)}, \ldots, \mathbf{H}^{(N)}$$

such that for any $n \in \{1, \ldots, N\}$,

$$\mathbf{H}^{(n)} \colon \mathbb{R}^{I_1 \times R} \times \cdots \times \mathbb{R}^{I_N \times R} \to \mathbb{R}^{I_n \times R}, \quad (\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}) \mapsto \mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)} - \mathbf{Z}_{(n)}\mathbf{\Phi}^{(n)},$$

where $\mathbf{\Phi}^{(n)}$ is given in (6.4). Then the fine-level problem can be formulated as a system of nonlinear equations

$$\mathbf{H}(\{\mathbf{A}\}) \coloneqq (\mathbf{H}^{(1)}(\{\mathbf{A}\}), \ldots, \mathbf{H}^{(N)}(\{\mathbf{A}\})) = (\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(N)}), \tag{6.21}$$

where $\mathbf{F}^{(n)} = \mathbf{0}$ for $n = 1, \ldots, N$ on the finest level. Note that we use $\{\mathbf{A}\}$ as shorthand for $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$. In order to apply the full approximation scheme we require a coarse version of (6.21). For each mode n we define the coarse operator

$$\mathbf{H}_c^{(n)}(\{\mathbf{A}_c\}) \coloneqq \mathbf{A}_c^{(n)}\mathbf{\Gamma}_c^{(n)} - \hat{\mathbf{Z}}_{(n)}^{c}\Big(\mathbf{A}_c^{(N)} \odot \cdots \odot \mathbf{A}_c^{(n+1)} \odot \mathbf{A}_c^{(n-1)} \odot \cdots \odot \mathbf{A}_c^{(1)}\Big), \tag{6.22}$$

where $\hat{\mathfrak{Z}}_c$ is the coarse-level tensor computed in the multiplicative setup phase. The coarse-level FAS equations are then given by

$$\mathbf{H}_c(\{\mathbf{A}_c\}) \coloneqq (\mathbf{H}_c^{(1)}(\{\mathbf{A}_c\}), \ldots, \mathbf{H}_c^{(N)}(\{\mathbf{A}_c\})) = (\mathbf{F}_c^{(1)}, \ldots, \mathbf{F}_c^{(N)}), \tag{6.23}$$

where

$$\mathbf{F}_c^{(n)} = \hat{\mathbf{R}}^{(n)}(\mathbf{F}^{(n)} - \mathbf{H}^{(n)}(\{\mathbf{A}\})) + \mathbf{H}_c^{(n)}(\{\tilde{\mathbf{A}}_c\}) \quad \text{for} \quad n = 1, \ldots, N \tag{6.24}$$

and $\hat{\mathbf{R}}^{(n)}$ is the mode-n restriction operator from the multiplicative setup phase. Here $\tilde{\mathbf{A}}_c^{(n)}$ is the coarse-level approximation of $\mathbf{A}^{(n)}$ obtained by restriction. Solving (6.23) for $\{\mathbf{A}_c\}$, the coarse-grid corrected factor matrices on the fine level are given by

$$\mathbf{A}_{\text{CGC}}^{(n)} = \mathbf{A}^{(n)} + \hat{\mathbf{P}}^{(n)}(\mathbf{A}_c^{(n)} - \tilde{\mathbf{A}}_c^{(n)}) \quad \text{for} \quad n = 1, \ldots, N, \tag{6.25}$$

where $\hat{\mathbf{P}}^{(n)}$ is the mode-n interpolation operator from the multiplicative setup phase. Together, (6.21) to (6.25) describe a FAS two-level coarse-grid correction scheme for the CP optimality equations.

6.3.2 Relaxation

We employ block nonlinear Gauss–Seidel (BNGS) as the relaxation scheme and coarsest-level solver for the CP-FAS algorithm (Algorithm 6.2). Applying BNGS to the equations in (6.21) is similar to applying ALS to the CP optimality equations. One iteration of BNGS consists of iterating through the modes sequentially, where at the nth step $\mathbf{\Gamma}^{(n)}$ and $\mathbf{\Phi}^{(n)}$ are computed and $\mathbf{A}^{(n)}$ is updated by solving

$$\mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)} = \mathbf{Z}_{(n)}\mathbf{\Phi}^{(n)} + \mathbf{F}^{(n)}. \tag{6.26}$$

When considering how to solve (6.26) for mode n, on any level, we note that an exact solution of the CP optimality equations is a fixed point of FAS only if it is a fixed point of the relaxation scheme. Suppose we update $\mathbf{A}^{(n)}$ by postmultiplying the right-hand side of (6.26) by $(\mathbf{\Gamma}^{(n)})^\dagger$, which is a small $R \times R$ matrix. If $\mathbf{\Gamma}^{(n)}$ is nonsingular, then its pseudoinverse is equivalent to its inverse, in which case there exists a unique solution and the fixed point is preserved. However, if $\mathbf{\Gamma}^{(n)}$ is singular, then postmultiplying by its pseudoinverse will in general not preserve the fixed point. Therefore, we propose using a few iterations of Gauss–Seidel (GS) to update $\mathbf{A}^{(n)}$, which guarantees the fixed-point property of our relaxation method. Moreover, a result by Keller [73] for positive semidefinite matrices states that if $\mathbf{\Gamma}^{(n)}$ has nonzero entries on its diagonal then GS must converge to a solution (there may be many) of (6.26). Owing to the structure of $\mathbf{\Gamma}^{(n)}$ this condition is equivalent to the fundamental requirement that the factor matrices have nonzero columns. Therefore, we can be confident that GS will converge regardless of whether or not $\mathbf{\Gamma}^{(n)}$ is singular. In practice we find that only a few GS iterations are necessary to obtain a sufficiently accurate solution to (6.26), and that further iterations do little to improve the relaxed approximation (our numerical tests use ten GS iterations).

Due to the structure of the FAS equations, in particular the right-hand side in (6.21), the scaling and permutation indeterminacies are not present on the coarser levels and so normalizing/reordering there is unnecessary. Therefore, normalization and reordering (as described in §6.1) are performed only on the finest level.

6.3.3 CP-FAS algorithm

A pseudocode description for an additive solve phase V-cycle is given in Algorithm 6.2. We assume that at any given level the current tensor, the index set $\mathcal{I}_c$, and the interpolation/restriction operators from the setup phase are available to the algorithm. Note that the parameters (ν_1, ν_2, ν_c) may be different from those used during the setup phase.

It is instructive to mention the differences between this additive solution phase and the additive phase in [47]. In the SVD case, singular vectors can be computed in separate V-cycles and FAS is not required because the singular values are updated in a top-level Ritz step. In the tensor case, all factor vectors need to be computed simultaneously in a single FAS V-cycle, and the weights λ_r from (6.7), which are in some sense equivalent to the singular values, are updated in these FAS cycles as well, making a Ritz step unnecessary.

Algorithm 6.2: V-cycle for solve phase of CP decomposition (CP-FAS)

```
   Input: right-hand side matrices $\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(N)}$, factor matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$
   Output: updated factor matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$
   if not on the coarsest level then
1.     Apply $\nu_1$ relaxations to $\mathbf{H}(\{\mathbf{A}\}) = (\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(N)})$
2.     Coarse initial guess:
           $\tilde{\mathbf{A}}_c^{(n)} \leftarrow \begin{cases} \hat{\mathbf{R}}^{(n)}\mathbf{A}^{(n)} & \text{if } n \in \mathcal{I}_c, \\ \mathbf{A}^{(n)} & \text{if } n \in \mathcal{I}_c' \end{cases} \quad \text{for} \quad n = 1, \ldots, N$
3.     Coarse right-hand side:
           $\mathbf{F}_c^{(n)} \leftarrow \begin{cases} \mathbf{H}_c^{(n)}(\{\tilde{\mathbf{A}}_c\}) + \hat{\mathbf{R}}^{(n)}(\mathbf{F}^{(n)} - \mathbf{H}^{(n)}(\{\mathbf{A}\})) & \text{if } n \in \mathcal{I}_c, \\ \mathbf{F}^{(n)} & \text{if } n \in \mathcal{I}_c' \end{cases} \quad \text{for} \quad n = 1, \ldots, N$
4.     Recursive solve:
           $\{\mathbf{A}_c^{(1)}, \ldots, \mathbf{A}_c^{(N)}\} \leftarrow \text{CP-FAS}(\mathbf{F}_c^{(1)}, \ldots, \mathbf{F}_c^{(N)},\ \tilde{\mathbf{A}}_c^{(1)}, \ldots, \tilde{\mathbf{A}}_c^{(N)})$
5.     Coarse-grid correction:
           $\mathbf{A}^{(n)} \leftarrow \mathbf{A}^{(n)} + \begin{cases} \hat{\mathbf{P}}^{(n)}(\mathbf{A}_c^{(n)} - \tilde{\mathbf{A}}_c^{(n)}) & \text{if } n \in \mathcal{I}_c, \\ \mathbf{A}_c^{(n)} - \tilde{\mathbf{A}}_c^{(n)} & \text{if } n \in \mathcal{I}_c' \end{cases} \quad \text{for} \quad n = 1, \ldots, N$
6.     Apply $\nu_2$ relaxations to $\mathbf{H}(\{\mathbf{A}\}) = (\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(N)})$
   else
7.     Apply $\nu_c$ relaxations to $\mathbf{H}(\{\mathbf{A}\}) = (\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(N)})$
   end
```

We conclude this section with a simple fixed-point theorem for the CP-FAS V-cycle. We note that the exact solution of the optimality equations is generally not a fixed point of the CP-AMG-mult V-cycle (especially when there are many components). However, if the exact solution is provided as the initial guess, then the setup phase typically produces an approximate solution that lies in the basin of attraction of the solution phase.

Theorem 6.3.1 (CP-FAS V-cycle fixed point property). *Let $\{\mathbf{A}\} = \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$ be a solution of the fine level CP optimality equations. Then $\{\mathbf{A}\}$ is a fixed point of the CP-FAS V-cycle.*

Proof. We prove this result for a two-level CP-FAS cycle; the proof can be extended to the multilevel case by induction over the levels. We begin by noting that $\{\mathbf{A}\}$ is a fixed point of the fine-level relaxations (see §6.3.2); therefore, it is sufficient to consider the CP-FAS two-level coarse-grid correction step. Since $\mathbf{H}^{(n)}(\{\mathbf{A}\}) = \mathbf{0}$ for $n = 1, \ldots, N$ it follows that $\mathbf{F}_c^{(n)} = \mathbf{H}_c^{(n)}(\{\tilde{\mathbf{A}}_c\})$ for $n = 1, \ldots, N$. Thus, the coarse-level equations are given by

$$\mathbf{H}_c^{(n)}(\{\mathbf{A}_c\}) = \mathbf{H}_c^{(n)}(\{\tilde{\mathbf{A}}_c\}) \quad \text{for} \quad n = 1, \ldots, N.$$

Using the BNGS relaxation scheme discussed in §6.3.2 with $\{\tilde{\mathbf{A}}_c\}$ as the initial guess to solve the coarse-level equations, the solution on the coarse level is given by $\mathbf{A}_c^{(n)} = \tilde{\mathbf{A}}_c^{(n)}$ for $n = 1, \ldots, N$. Computing the coarse-grid correction we find that

$$\mathbf{A}_{\text{CGC}}^{(n)} = \mathbf{A}^{(n)} + \hat{\mathbf{P}}^{(n)}(\mathbf{A}_c^{(n)} - \tilde{\mathbf{A}}_c^{(n)}) = \mathbf{A}^{(n)} \quad \text{for} \quad n = 1, \ldots, N.$$

□

6.3.4 Full multigrid FAS cycles

For some tensors the initial guess provided by the multiplicative setup phase may be inadequate to yield a convergent additive solve phase, that is, the initial guess lies outside the basin of attraction. One way in which we can try to obtain a better initial guess to the fine-level problem is to use full multigrid (see §2.6.5). Once an initial guess to the finest-level problem has been obtained we can apply repeated CP-FAS cycles to obtain an improved approximate solution. We use CP-FAS V-cycles as the solver on each level of the FMG cycle, except on the coarsest level, where ALS is used (see §6.1). A pseudocode description of the FMG-CP-FAS algorithm is given in Algorithm 6.3. We assume that at any given level the current tensor, the index set $\mathcal{I}_c$, and the interpolation/restriction operators from the setup phase are available. In Algorithm 6.3 we use a subscript ℓ to index the current level, where $\ell = 0, \dots, L$. Note that a subscript ℓ on an interpolation operator indicates that level ℓ is mapped to level $\ell - 1$.

Algorithm 6.3: FMG cycle for solve phase of CP decomposition (FMG-CP-FAS)

```
   Output: finest-level factor matrices $\mathbf{A}_0^{(1)}, \dots, \mathbf{A}_0^{(N)}$
1. On the coarsest level apply $\nu$ iterations of ALS with a random initial guess to obtain
   $\mathbf{A}_L^{(1)}, \dots, \mathbf{A}_L^{(N)}$
2. Set $\ell \leftarrow L - 1$
   while $\ell \neq 0$ do
3.     $\mathbf{A}_{\ell-1}^{(n)} \leftarrow \hat{\mathbf{P}}_\ell^{(n)} \mathbf{A}_\ell^{(n)}$ for $n = 1, \dots, N$
4.     $\{\mathbf{A}_{\ell-1}^{(1)}, \dots, \mathbf{A}_{\ell-1}^{(N)}\} \leftarrow$ CP-FAS($\{\mathbf{0}\}$, $\mathbf{A}_{\ell-1}^{(1)}, \dots, \mathbf{A}_{\ell-1}^{(N)}$)
5.     $\ell \leftarrow \ell - 1$
   end
```

6.4 Implementation details and numerical results

In this section we present the results of numerical tests. All experiments are performed using MATLAB version 7.5.0.342 (R2007b) and version 2.4 of the Tensor Toolbox [3]. Timings are reported for a laptop running Windows XP, with a 2.50 GHz Intel Core 2 Duo processor and 4 GB of RAM. Initial guesses for the boot factors and test factors are randomly generated from the standard uniform distribution. The initial boot factors are also used as the initial guess for the standalone ALS method. The stopping criterion for the numerical tests is based on the gradient of f. In particular, defining

$$g(\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}) := \frac{1}{\|\mathfrak{Z}\|} \left(\sum_{n=1}^{N} \|\mathbf{G}^{(n)}\|^2 \right)^{1/2}, \tag{6.27}$$

where $\mathbf{G}^{(n)}$ is the mode-n partial derivative of f as defined in (6.3), we iterate until

$$g(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}) < \tau \tag{6.28}$$

or until the maximum number of iterations is reached. For the multilevel method the maximum number of iterations is set to 500. For ALS the maximum number of iterations is set to 10^4. The stopping tolerance is $\tau = 10^{-10}$. Table 6.1 lists the parameters used by the setup and solve phases. As in [47, 72, 83], a larger number of relaxations is required in the setup cycles to produce sufficiently accurate transfer operators.

Table 6.1: CP-AMG-mult and CP-FAS parameters.

Parameter	CP-AMG-mult	CP-FAS
Pre-relaxations ν_1	5	1
Post-relaxations ν_2	5	1
Relaxations on coarsest level ν_c	100	50
Cycle type	V-cycle	V-cycle
Number of test blocks n_t	2	n/a

For each numerical test, we perform ten runs with a different random initial guess for each run. The values reported in the tables represent averages over the successful runs, where a run is deemed successful if the stopping criterion is satisfied prior to reaching the iteration limit. The tables compare the ALS method and the multilevel method with or without FMG-CP-FAS as part of the setup phase (see §6.4.1). For ALS we report the average number of iterations, the average execution time, and the number of successful runs. For the multilevel method we report the average number of iterations (setup and solve phases), the average total execution time, the number of successful runs, the average speedup over ALS, and the number of levels. The average speedup is determined as follows. For a given test and run, if both ALS and the multilevel method were successful, we divide the execution time of ALS by the execution time of the multilevel method to obtain the speedup for that run. The speedup values for all runs are then averaged to obtain the average speedup for that test. We note that execution times do not include the evaluation of the stopping criterion, that is, the computation of g in (6.27).

6.4.1 Implementation details

The multilevel setup and solve phases have thus far been described separately; however, these phases can be combined in the following simple way. Since the factor matrices lie only approximately in the range of the interpolation operators, convergence of the setup cycles, as measured by the functional g, should stagnate after a few iterations. Therefore, after each setup cycle the current iterate $\{\mathbf{A}_{new}\}$ is compared to the previous iterate $\{\mathbf{A}_{old}\}$, and the setup cycles are halted once

$$g(\{\mathbf{A}_{new}\}) > (1-\varepsilon)g(\{\mathbf{A}_{old}\}), \tag{6.29}$$

where the tolerance is set at $\varepsilon = 0.1$. At most five setup cycles are performed, and stopping criterion (6.29) is checked only after three setup cycles have completed. After the setup phase is complete, solution cycles are performed until the stopping criterion (6.28) is satisfied. To improve robustness, we also try to detect stagnation of the solution cycles. After five solution cycles have elapsed, the stagnation condition $g(\{\mathbf{A}_{new}\}) \geq g(\{\mathbf{A}_{old}\})$ is checked in each subsequent iteration. If this inequality is satisfied then the current iterate $\{\mathbf{A}_{new}\}$ is discarded, and the transfer operators and coarse tensors are rebuilt by one downsweep of CP-AMG-mult with the previous iterate $\{\mathbf{A}_{old}\}$ used for the boot factors. This process is carried out at most once, and any further indications of stagnation are ignored. We note that the boot factors are not updated by the downsweep of CP-AMG-mult, as doing so would likely ruin any progress made by the solution cycles.

The combination of the setup and solve phases described above can be modified to include an FMG-CP-FAS cycle as part of the setup phase. After the setup cycles have completed, we perform one FMG-CP-FAS cycle to compute a new approximation to the boot factors. The transfer operators and coarse tensors are then rebuilt using one downsweep of CP-AMG-mult. Note that while the TFMs are updated by the downsweep of CP-AMG-mult, the boot factors are not. We refer to this combination as "Multilevel +

FMG" in the tables and "ML + FMG" in the figures.

We conclude this section by considering the computational costs of one setup cycle, one solution cycle, and one FMG cycle. Let $\ell = 0, \dots, L$ index the levels, and define

$$I_n^\ell := \frac{I_n}{2^\ell}, \qquad P^\ell := \prod_{n=1}^{N} I_n^\ell = \frac{1}{2^{N\ell}} \prod_{n=1}^{N} I_n, \qquad S^\ell := \sum_{n=1}^{N} I_n^\ell = \frac{1}{2^\ell} \sum_{n=1}^{N} I_n$$

for any $\ell \geq 0$. Assume for simplicity that $\boldsymbol{\mathfrak{Z}}$ is dense and that each mode is coarsened at the same rate with L being the same for each mode. Consideration of Algorithm 6.1 shows that the most expensive operations on each level are the construction of the coarse-level tensor, the relaxations, and the construction of the interpolation operators, in particular computing the weights for the least squares fits. We note that since $\mathbf{B}^{(n)}$ is tridiagonal (due to the structure of the interpolation matrices), its Cholesky factor can be computed in only $\mathcal{O}(S^\ell/2)$ operations on the ℓth level. The coarse-level tensor is constructed by sequentially taking the n-mode product of the current tensor with the nth restriction operator for $n = 1, \dots, N$. Computing $\hat{\boldsymbol{\mathfrak{Z}}}^c$ on level ℓ requires $\mathcal{O}(P^\ell S^\ell)$ operations. The dominant computation for the relaxations and least squares weights is the matrix product $\mathbf{Z}_{(n)}\mathbf{\Phi}^{(n)}$. Since $\mathbf{Z}_{(n)}$ is of size $I_n^\ell \times (P^\ell/I_n^\ell)$ and $\mathbf{\Phi}^{(n)}$ is of size $(P^\ell/I_n^\ell) \times R$ on the ℓth level, forming this product requires $\mathcal{O}(NP^\ell R)$ operations. Therefore, by summing over all the levels, to leading order one setup cycle requires approximately

$$\left[\left(\frac{2^N}{2^N - 1}\right)(n_t(\nu_1 + 1) + \nu_1 + \nu_2 + 1) + \frac{\nu_c}{2^{NL}}\right] \cdot \mathcal{O}(NPR) + \left(\frac{2^N}{2^N - 1}\right) \cdot \mathcal{O}(PS) \tag{6.30}$$

operations, where $P = P^0$ and $S = S^0$. We note that PS scales only slightly worse than linear in P, and in particular $NPI_{min} \leq PS \leq NPI_{max}$, where I_{min} and I_{max} are the sizes of the smallest and largest modes, respectively. Consideration of Algorithm 6.2 shows that the most expensive operations on each level are the relaxations and the construction of the right-hand sides. By a similar analysis, to leading order one solution cycle requires approximately

$$\left[\left(\frac{2^N}{2^N-1}\right)(\nu_1+\nu_2+1+1/2^N)+\frac{\nu_c}{2^{NL}}\right]\cdot\mathcal{O}(NPR) \tag{6.31}$$

operations. Similarly, it follows that to leading order one FMG-CP-FAS cycle requires approximately

$$\left[\left(\frac{2^N}{2^N-1}\right)^2(\nu_1+\nu_2+1+1/2^N)+\frac{\nu+\nu_c L}{2^{NL}}\right]\cdot\mathcal{O}(NPR) \tag{6.32}$$

operations. Note that in (6.32) ν is the number of ALS iterations performed on the coarsest level and $(\nu_1,\ \nu_2,\ \nu_c)$ are the CP-FAS parameters. In general a solution cycle is significantly cheaper than a setup cycle because of the extra work required by a setup cycle to relax on the TFMs, the typically larger number of relaxations performed on each level of a setup cycle, and the added work of constructing the coarse-level tensors (i.e., the $\mathcal{O}(PS)$ term). If $\mathfrak{Z}$ is sparse then further savings are possible on the finest level. In particular, to leading order the cost of one relaxation reduces to NR times the number of nonzero elements in $\mathfrak{Z}$. In our current framework the coarse tensors will in general be dense; multiplication by the inverted Cholesky factors as in (6.12) eliminates any sparsity. Therefore, it may be interesting to consider alternative formulations of the coarse-level equations without multiplication by Cholesky factors, for example, by working directly with the coarse-level equations (6.10). In the analogous case of computing SVD triplets of a matrix [47] via adaptive algebraic multigrid the coarse-level equations correspond to a generalized singular value problem that can be relaxed by block Gauss–Seidel and solved directly on the coarsest level. In our case, however, it is currently unclear if the coarse-level equations without the Cholesky factors (6.10) can be interpreted as the optimality equations of a *generalized canonical decomposition*, and if ALS can be adapted as a relaxation scheme for these equations.

6.4.2 Sparse tensor test problem

The first test problem we consider is the standard finite difference Laplacian tensor on a uniform grid of size s^d in d dimensions. This test problem yields an N-mode sparse tensor $\mathfrak{Z}$ of size $s \times s \times \cdots \times s$ with $N = 2d$. We can efficiently construct $\mathfrak{Z}$ by reshaping the $s^d \times s^d$ matrix

$$\mathbf{Z} = \sum_{k=1}^{d} \mathbf{I}_{\ell(k)} \otimes \mathbf{D} \otimes \mathbf{I}_{r(k)} \,,$$

where $\mathbf{I}_{r(k)}$ is the $s^{k-1} \times s^{k-1}$ identity matrix, $\mathbf{I}_{\ell(k)}$ is the $s^{d-k} \times s^{d-k}$ identity matrix, and $\mathbf{D}$ is the $s \times s$ tridiagonal matrix with stencil $[-1,\, 2,\, -1]$. Although this test problem is somewhat pedagogical in nature, it offers a good starting point to illustrate our method. The parameters for the various test tensors that we consider are given in Table 6.2. Numerical results for the sparse problem are given in Table 6.3. The results show that our multilevel approach is anywhere from two to seven times as fast as ALS for this test problem. For tests 1 to 6 (order 4 tensors), larger speedups are observed for the multilevel method with FMG. However, for tests 7 to 11 (order 6 and 8 tensors) larger speedups are observed for the multilevel method without FMG. For higher-order tensors the setup phase of the multilevel method with FMG is considerably more expensive than the setup phase of the multilevel method without FMG. The multilevel variants demonstrate similar robustness to varying initial guesses for this problem; however, in general we expect the multilevel method with FMG to be the most robust option. We also observe the trend that for each grouping of tests in Table 6.3, the speedup tends to increase as the number of components R increases. Figures 6.2 and 6.3 illustrate the convergence history of ALS and the multilevel method for one run of tests 3 and 9, respectively, in Table 6.3. These plots are typical of the performance observed for this test problem. We note that the spike in the "Multilevel + FMG" curves is due to the initial approximation to the solution computed by the single FMG-CP-FAS cycle performed after the setup cycles.

Test	Problem parameters
1	$N = 4,\ s = 20,\ R = 4$
2	$N = 4,\ s = 20,\ R = 5$
3	$N = 4,\ s = 20,\ R = 6$
4	$N = 4,\ s = 50,\ R = 2$
5	$N = 4,\ s = 50,\ R = 3$
6	$N = 4,\ s = 50,\ R = 4$
7	$N = 6,\ s = 20,\ R = 2$
8	$N = 6,\ s = 20,\ R = 3$
9	$N = 6,\ s = 20,\ R = 4$
10	$N = 8,\ s = 10,\ R = 2$
11	$N = 8,\ s = 10,\ R = 3$

Table 6.2: Test parameters for sparse test problem.

	ALS			Multilevel				Multilevel + FMG				
test	it	time	ns	it	time	spd	ns	it	time	spd	ns	levs
1	1897	26.2	10	37	8.7	$3.0(0,0)$	10	36	9.2	$3.0(0,0)$	10	2
2	3329	54.1	8	64	15.6	$4.4(0,0)$	10	42	11.8	$5.0(0,0)$	10	2
3	3587	71.1	9	67	18.1	$3.9(0,0)$	9	32	10.6	$6.8(0,0)$	10	2
4	5457	98.9	10	113	37.9	$2.6(3,0)$	10	118	41.1	$2.5(0,0)$	10	4
5	5508	164.0	4	200	69.8	$2.5(5,1)$	9	100	43.1	$3.8(1,1)$	9	4
6	6788	247.8	3	163	71.2	$5.1(2,0)$	10	146	68.1	$5.2(2,0)$	10	4
7	1619	227.4	10	51	121.1	$1.9(0,0)$	10	52	135.9	$1.7(0,0)$	10	3
8	3481	718.6	10	72	185.7	$3.9(0,0)$	10	70	198.5	$3.7(0,0)$	10	3
9	4085	1137.9	10	75	237.0	$4.8(2,0)$	10	78	260.0	$4.5(1,0)$	10	3
10	634	227.4	10	50	180.1	$1.3(0,0)$	10	54	207.1	$1.1(0,0)$	10	3
11	1743	949.5	10	39	426.0	$2.2(0,0)$	10	43	498.8	$1.9(0,0)$	10	3

Table 6.3: Sparse problem. Average number of iterations (it) and time in seconds (time) until the stopping criterion is satisfied with stopping tolerance 10^{-10}. Here "spd" is the multilevel speedup compared to ALS and "ns" is the number of successful runs. The ordered pair (a, b) in the "spd" column gives the number of runs in which the transfer operators were rebuilt and the number of runs in which rebuilding the transfer operators failed to recover convergence, respectively.

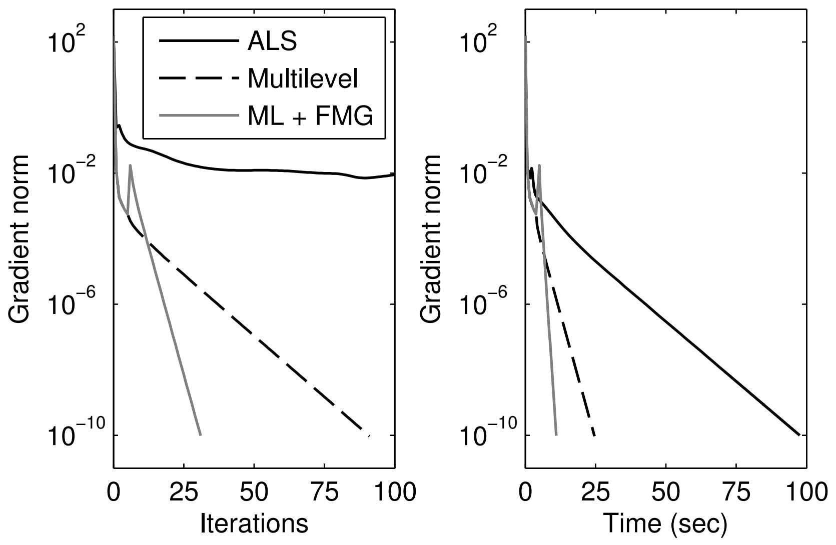


Figure 6.2: Sparse problem. Convergence plot for test 3 from Table 6.2 ($N = 4$, $s = 20$, $R = 6$).

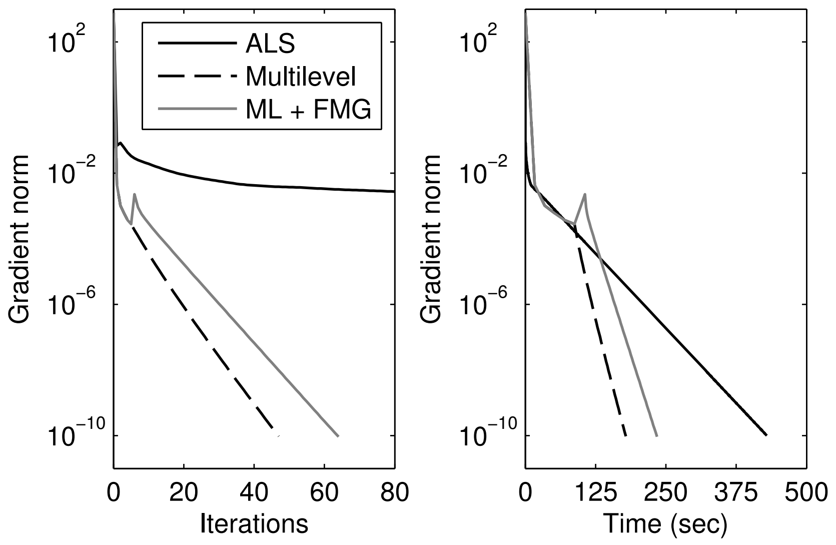


Figure 6.3: Sparse problem. Convergence plot for test 9 from Table 6.2 ($N = 6$, $s = 20$, $R = 4$).

6.4.3 Dense tensor test problem

The second test problem we consider is a dense, symmetric, third-order tensor $\mathfrak{Z} \in \mathbb{R}^{s\times s\times s}$ whose elements are given by

$$z_{ijk} = \left(i^2 + j^2 + k^2\right)^{-1/2} \quad \text{for} \quad i, j, k = 1, \ldots, s.$$

This tensor was used as a test case in [97], and was also considered in [98], which describes a novel method for computing the Tucker decomposition of third-order tensors. As mentioned in [98], $\mathfrak{Z}$ arises from the numerical approximation of an integral equation with kernel $1/\|\mathbf{x} - \mathbf{y}\|$ acting on the unit cube and discretized by the Nyström method on a uniform grid. In this section we compute CP decompositions for $R = 2$, 3, 4, 5. It has been observed numerically that when $R \geq 4$ ALS may be extremely slow to converge, requiring on the order of 10^5 iterations for some initial guesses, with highly nonmonotonic convergence behavior as measured by g in (6.27). The performance of our method for $R \geq 4$ is less robust than desired because the multigrid framework uses a single interpolation operator for each factor matrix. Even so, depending on the initial guess our method may still demonstrate a significant improvement over ALS.

The results for the dense problem are given in Table 6.4. We note that only the multilevel method with FMG is considered (see the description in §6.4.1). The blank entries for test 4 in Table 6.4 indicate that ALS did not have any successful runs. For $R \geq 3$ our multilevel approach can lead to significant savings in iterations and execution time. The speedup is less impressive when $R = 2$ because ALS already converges quickly without any multigrid acceleration. It is also apparent that as the number of components increases, the number of successful runs of the multilevel method, and of ALS, decreases. For initial guesses in which the multilevel method failed to converge, there was typically a rapid decrease in the gradient norm, followed by convergence stagnation of the solution cycles. This behavior suggests that the setup phase was unable to construct transfer operators that adequately represented the solution in their range. Alternatively, it is possible that ALS may have entered a *swamp* which is essentially a protracted region of slow error reduction. Swamps are artifacts of the ALS procedure typically characterized by highly collinear factors in all modes (see §6.4.4 for the definition of collinearity). How to deal with swamps in ALS is an open topic of research, and currently our algorithm does not implement any special procedures to deal with swamps in ALS (see [120] and the references within).

		ALS			Multilevel + FMG				
test	problem parameters	it	time	ns	it	time	spd	ns	levs
1	$s = 50$, $R = 2$	161	0.7	10	7	2.0	0.4(0, 0)	10	5
2	$s = 50$, $R = 3$	2435	11.8	10	10	2.0	5.7(1, 0)	10	5
3	$s = 50$, $R = 4$	4838	25.9	5	140	13.4	3.7(8, 2)	8	4
4	$s = 50$, $R = 5$	—	—	0	276	27.2	—(9, 6)	3	4
5	$s = 100$, $R = 2$	253	10.5	10	7	7.2	1.5(0, 0)	10	6
6	$s = 100$, $R = 3$	1695	78.8	9	9	7.8	10.2(0, 0)	10	6
7	$s = 100$, $R = 4$	3836	198.5	6	125	38.2	13.6(8, 0)	10	5
8	$s = 100$, $R = 5$	7854	437.9	2	219	68.3	9.2(10, 4)	6	5
9	$s = 200$, $R = 2$	274	88.2	10	7	46.9	1.9(0, 0)	10	7
10	$s = 200$, $R = 3$	1830	663.1	10	16	66.9	10.5(2, 0)	10	7
11	$s = 200$, $R = 4$	2998	1209.1	8	80	179.9	9.7(8, 1)	9	6
12	$s = 200$, $R = 5$	5686	2529.0	3	209	421.6	5.9(9, 6)	4	6

Table 6.4: Dense problem. Average number of iterations (it) and time in seconds (time) until the stopping criterion is satisfied with stopping tolerance 10^{-10}. Here "spd" is the multilevel speedup compared to ALS and "ns" is the number of successful runs. The ordered pair (a, b) in the "spd" column gives the number of runs in which the transfer operators were rebuilt and the number of runs in which rebuilding the transfer operators failed to recover convergence, respectively.

Figures 6.4, and 6.5 illustrate the convergence history of ALS and the multilevel method for one run of tests 7, and 8, respectively, in Table 6.4. Figure 6.5 ($R = 5$) shows how ALS can initially be slow to converge with erratic convergence behavior: for the first half of the run its gradient norm fluctuates with little decrease. Such behavior can make it very difficult for the setup phase to construct adequate transfer operators.

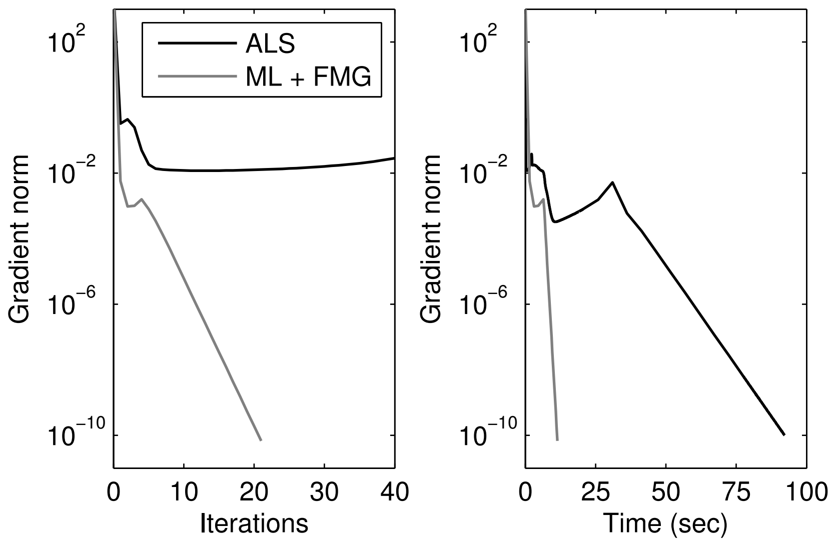


Figure 6.4: Dense problem. Convergence plot for test 7 from Table 6.4 ($s = 100$, $R = 4$).

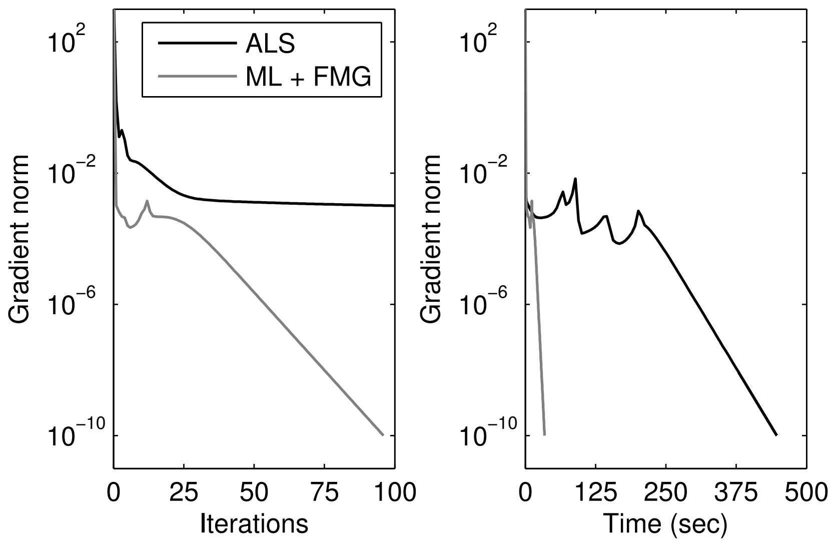


Figure 6.5: Dense problem. Convergence plot for test 8 from Table 6.4 ($s = 100$, $R = 5$).

6.4.4 Random data test problem

As our final test problem we consider factoring third-order random data tensors $\mathfrak{Z} \in \mathbb{R}^{s\times s\times s}$ of ranks $R = 3, 4, 5$. A test tensor is constructed by randomly generating factor matrices $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)} \in \mathbb{R}^{s\times R}$ such that the *collinearity* c of the factor matrices is specified in advance. This is a standard non-PDE test problem for CP decomposition that was considered in [1, 116], and also more recently in [46]. Setting the collinearity of the factor vectors to c means that

$$\frac{\left\langle \mathbf{a}_r^{(n)}, \mathbf{a}_q^{(n)} \right\rangle}{\|\mathbf{a}_r^{(n)}\| \cdot \|\mathbf{a}_q^{(n)}\|} = c \quad \text{for} \quad q \neq r, \quad r, q = 1, \ldots, R, \quad \text{and} \quad n = 1, 2, 3. \tag{6.33}$$

We set $c = 0.9$ for our test problems because it is well-known that collinearity of the factors near unity leads to slow convergence of ALS [116].

The following steps are used to generate a third-order test tensor $\mathfrak{Z}$ with rank R and collinearity c.

1. Generate an $R \times R$ matrix $\mathbf{C}$ that has ones on its diagonal and off-diagonal elements equal to c, and compute its Cholesky factor $\mathbf{L}$.

2. Generate three uniformly random $s \times R$ matrices, $\tilde{\mathbf{A}}^{(1)}, \tilde{\mathbf{A}}^{(2)}, \tilde{\mathbf{A}}^{(3)}$ and orthonormalize their columns via QR factorization:

$$\tilde{\mathbf{A}}^{(n)} = \mathbf{Q}^{(n)}\mathbf{R}^{(n)} \quad \text{for} \quad n = 1, 2, 3.$$

3. Set $\mathbf{A}^{(n)} = \mathbf{Q}^{(n)}\mathbf{L}$ for $n = 1, 2, 3$ and let $\mathfrak{Z} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}]\!]$.

We note that in [1, 116] two types of noise are added to the test tensors; however, we do not consider the addition of noise in our test cases. The results for the collinear test problem are presented in Table 6.5. It is clear that ALS is slow to converge (because of the high level of collinearity between the factors), and our multilevel approach can lead to significant savings in both iterations and execution time. Unfortunately, our method appears to be less robust

		ALS			Multilevel				
test	problem parameters	it	time	ns	it	time	spd	ns	levs
1	$s = 100,\ R = 3,\ c = 0.9$	1931	90.1	10	55	20.1	$4.5(3, 2)$	8	5
2	$s = 100,\ R = 4,\ c = 0.9$	2286	118.9	10	115	37.9	$4.2(5, 4)$	6	5
3	$s = 100,\ R = 5,\ c = 0.9$	2576	147.3	10	78	31.5	$4.7(8, 3)$	7	5
4	$s = 200,\ R = 3,\ c = 0.9$	1892	697.1	10	54	136.6	$5.2(3, 2)$	8	6
5	$s = 200,\ R = 4,\ c = 0.9$	2266	931.1	10	67	177.5	$5.3(6, 4)$	6	6
6	$s = 200,\ R = 5,\ c = 0.9$	2613	1185.2	10	77	214.6	$5.4(8, 6)$	4	6

Table 6.5: Random data problem. Average number of iterations (it) and time in seconds (time) until the stopping criterion is satisfied with stopping tolerance 10^{-10}. Here "spd" is the multilevel speedup compared to ALS and "ns" is the number of successful runs. The ordered pair (a, b) in the "spd" column gives the number of runs in which the transfer operators were rebuilt and the number of runs in which rebuilding the transfer operators failed to recover convergence, respectively.

than ALS, which converges for each run, albeit quite slowly. We have found that using FMG as part of the setup phase does not improve robustness for this test problem. The observed lack of robustness is not surprising because we are using AMG components that were developed for PDE applications. In particular, it is well-known that applying AMG to new classes of problems often requires new coarsening techniques. We use simple geometric coarsening (every other fine-level point becomes a coarse-level point), and fine-level points interpolate from their lexicographically nearest neighbors. This coarsening is somewhat arbitrary because there is no immediate reason to expect the random data to be correlated between neighboring points, and it is clear that more sophisticated coarsening techniques are required to obtain robust results. For example, it may be possible to remedy these robustness issues by employing a strength-based coarsening procedure similar to that used in [47], and this idea is currently the focus of ongoing research. Nevertheless, the results in Table 6.5 clearly demonstrate that our approach is promising for non-PDE type problems, with the potential for significant speedups. This observation is supported in Figures 6.6 and 6.7, which illustrate the convergence history of ALS and our multilevel method for one run of tests 3 and 5, respectively, in Table 6.5.

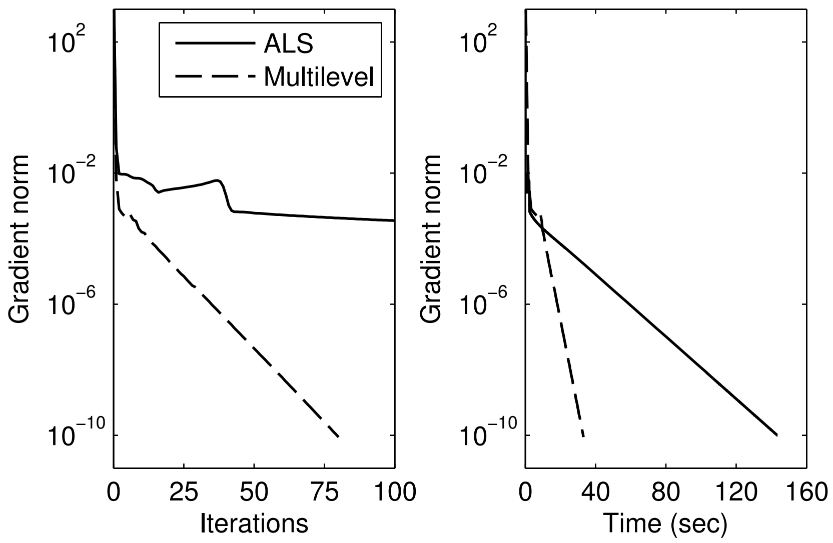


Figure 6.6: Random data problem. Convergence plot for test 6 from Table 6.5 ($s = 100$, $R = 5$).

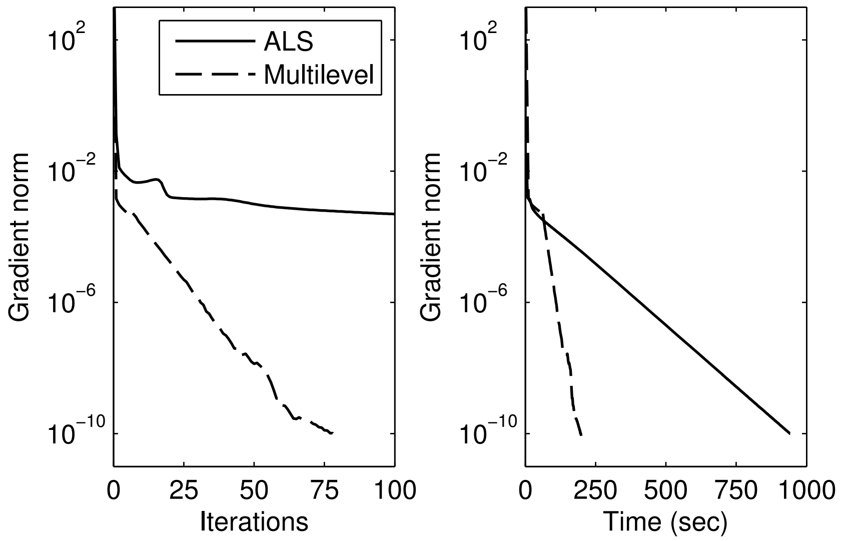


Figure 6.7: Random data problem. Convergence plot for test 9 from Table 6.5 ($s = 200$, $R = 4$).

6.5 General discussion and conclusions

The main contribution of this chapter was to show how techniques from adaptive multigrid could be used to accelerate the alternating least squares method for computing the CP decomposition with small number of components. As far as we are aware, our method is the first genuine multigrid algorithm for computing the CP decomposition as well as the first adaptive AMG method for solving a nonlinear optimization problem. Numerical tests with dense and sparse tensors of varying sizes and orders (up to order eight) that are related to PDE problems showed how our multilevel method can lead to significant speedup over standalone ALS when high accuracy is desired. Furthermore, a test case that is unrelated to PDE problems demonstrated how our multilevel method may be successfully applied to more general test problems.

The main area in which our algorithm can be improved is its robustness. The often erratic convergence behavior of the ALS method can make it difficult to construct sufficiently accurate transfer operators during the setup phase. Therefore, it may be beneficial to consider a Tikhonov regularized formulation of CP [1]. As discussed in [120], the regularized problem is always well-posed in that it admits a global minimizer. Moreover, numerical investigation has shown that *swamps* in ALS convergence (protracted regions of very slow error reduction) are less likely to occur with the regularized formulation. A more general coarsening routine would also lead to improved robustness in that a wider class of tensors could be considered. As in classical AMG, a rigorous (at the very least heuristic) characterization of the error for certain classes of tensors will be key in guiding the development of an automatic operator-based coarsening routine. It may also be worthwhile to investigate a more sophisticated setup phase that iteratively combines the CP-AMG-mult cycles and CP-FAS-FMG cycles, and more sophisticated measures of the convergence stagnation. How best to deal with stagnation is yet another question. As discussed in [47], the addition of a line search on the finest level may also improve convergence and robustness.

Further avenues of research include identifying classes of tensors for which multigrid acceleration of ALS may be beneficial, developing an alternative formulation of the coarse-level equations without the inverted Cholesky factors, and generalizing our multilevel framework to other similar tensor optimization problems, such as the Tucker decomposition [79], and block tensor decompositions [41, 44], as well as the best rank-$(R_1, \ldots, R_N)$ approximations [42, 70].

Chapter 7

Conclusions and Future Work

The objective of this thesis was to develop efficient and robust numerical methods based on adaptive algebraic multigrid for solving problems in linear and multilinear algebra. In particular, we have considered two relevant applications, namely, computing the stationary distribution of large sparse Markov chains, and computing the canonical decomposition of higher-order tensors. Numerical methods have been motivated through heuristic-based arguments, and have been validated through extensive numerical experimentation. In the following sections we summarize the major results of this thesis and discuss avenues of future research.

7.1 Markov chains

7.1.1 Contributions

In Chapter 3 we presented a novel algorithm for Markov chains that combines classical algebraic multigrid techniques with an exact interpolation scheme framework. By adding the computed solution directly to the range of interpolation, as in the exact interpolation scheme, the approximation of near-nullspace components by the range of interpolation was adaptively improved as the computed solution converged. Through a heuristic analysis we showed that algebraically smooth multiplicative error is locally constant along strong connections in the system operator scaled by the relaxed approximation, which motivated the use of classical AMG coarsening and interpolation. In order for our heuristic analysis to apply on all levels it was necessary to maintain the irreducible singular M-matrix structure of the coarse-level system operators on all levels. By employing an existing lumping technique that augments the Galerkin coarse-level system operators through a small additive perturbation, the singular M-matrix structure of the fine-level system operator was preserved. Although this lumping technique is somewhat artificial, it was the only way in which we were able to guarantee the structure of the coarse-level system operators and strict positivity of the computed solution. The MCAMG algorithm (Algorithm 3.1) was vetted through a series of challenging test problems, for which it demonstrated excellent robustness, and near-optimal scalability. Although these results are encouraging, the MCAMG algorithm suffers from the fundamental drawback that the entire multilevel hierarchy of operators is recomputed during each multilevel cycle. To remedy this issue we developed a simple hybrid method in which the MCAMG method formed the setup phase, and a cheap additive correction multigrid method with a fixed multigrid hierarchy was used for the solve phase. Compared with the standalone MCAMG algorithm, the hybrid approach consistently resulted in faster execution times, and often lead to significant speedups. In general, we expect that more sophisticated hybrid schemes such as the parallel *on-the-fly* scheme proposed in [118] would lead to further improvements.

In Chapter 4 we proposed a constrained iterant recombination approach to accelerate multilevel methods for Markov chains. Although our numerical experiments focused on accelerating the simple adaptive multilevel aggregation method for Markov chains described in §4.1, our approach can be applied to any iterative method for Markov chains whose iterates are strictly positive probability vectors. The minimization problem for selecting the weights in the "optimal" linear combination of previous iterates consisted of minimizing the one-norm or the two-norm of the residual subject to strict positivity constraints on recombined iterate. Numerical results demonstrated that recombining only a small number of iterates (small window size) may be sufficient to significantly improve iteration counts as well as the scalability in the case of suboptimal performance of the standalone method. Moreover, even when the standalone method was performing optimally in terms of scalability, iterant recombination was able to further reduce the iteration counts. It was observed that iteration counts are not further reduced by a significant amount for window sizes larger than two, and in some cases the added overhead of solving a larger minimization problem lead to increased execution times. In terms of iteration reduction, no significant difference was observed between the one-norm and two-norm minimization approaches. In terms of overall execution time, acceleration with window size two was the clear winner for the problem sizes considered, with the analytic solution method for two-norm minimization discussed in §4.3 resulting in the fastest overall execution times for all test problems. We note that in light of the results in Chapter 5, for sufficiently large problems it may be beneficial to use a window size greater than two. In essence, the added numerical cost of solving a larger minimization problem is outweighed by the added savings of performing fewer multilevel iterations.

In Chapter 5 we demonstrated that a simple over-correction approach can significantly improve the convergence of the AGG method for Markov chains described in §4.1. In particular, we developed an automatic over-correction mechanism in which the over-correction parameter α is computed on each level as the solution of a unconstrained minimization problem. Numerical tests demonstrated that fixed over-correction, in which a single fixed value of α is used on all levels and for all cycles, can dramatically improve the convergence of multilevel aggregation, often resulting in optimal or near-optimal performance with respect to algorithmic scalability. The downside of the fixed over-correction approach is that the over-correction parameter has to be selected through a costly a priori trial-and-error procedure. Moreover, judging by the variability of the fixed over-correction parameter about the value $\alpha = 2$, estimating a suitable fixed over-correction parameter is problem dependent and far from straightforward. It was also shown that automatic over-correction is capable of improving scalability and reducing execution times of multilevel aggregation. Unfortunately, it was not as robust as desired, and was unable to reduce iteration counts by the same amount as the fixed over-correction approach with a carefully chosen α. Numerical tests also compared the MCAMG method (Chapter 3) and multilevel aggregation accelerated by iterant recombination (Chapter 4) with popular Krylov subspace methods for Markov chains including ILU-preconditioned Bi-CGStab and GMRES. For the smaller problem sizes Bi-CGStab and GMRES were generally the fastest solvers or were very competitive; however, our multilevel approach was competitive and often superior to Bi-CGStab and GMRES for the larger problems . Although it is difficult to form general conclusions based on comparisons with a limited number of test problems, it should be noted that the MCAMG and IR-AGG methods consistently showed good scalability and robustness, and were among the fastest multilevel methods for each of the test problems considered. Preliminary tests suggest that combining iterant recombination and over-correction may lead to further improvements over the individual acceleration approaches. In particular, iterant recombination in conjunction with fixed over-correction seems promising, although further testing is required.

7.1.2 Future research

With respect to the MCAMG algorithm, future research should aim at gaining further insight into the multiplicative correction scheme framework presented in this thesis. Many of the arguments in support of this framework are based on heuristics, and the goal should be to find mathematically rigorous justifications of these heuristics. In particular, a characterization of algebraically smooth error arising from weighted Jacobi relaxation applied to irreducible singular M-matrices would provide insight into the construction of coarse grids and the definition of interpolation. Since the energy norm cannot be used for Markov chain problems, one of the first steps toward a characterization of algebraically smooth error is try and find a suitable norm on which to base our analysis. Additional areas of investigation include an iteration-dependent strength threshold that adapts with respect to the measured operator complexity, alternative techniques to reduce the operator complexity, a more rigorous theoretical investigation of the lumped coarse-level system, and the efficient parallel implementation of MCAMG to facilitate large-scale testing.

With respect to iterant recombination acceleration of multilevel aggregation, it may be worthwhile to investigate constrained optimization solvers other than the ellipsoid method for solving the one-norm minimization problem. As discussed in §4.6, the one-norm minimization problem is equivalent to a linear programming problem, for which interior point methods and the simplex method are known to be effective solvers.

With respect to over-correction of multilevel aggregation, future research should focus on improving the robustness and performance of the automatic over-correction approach. A rigorous analysis of the over-correction mechanism, or at the very least a heuristic analysis involving a suitable model problem, would be of great benefit to guide the formulation of the automatic over-correction minimization problem. In addition, it may be sufficient to automatically determine the over-correction parameter on the finest level, and then fix this value on coarser levels. While this approach clearly improves the per iteration cost of over-correction, it may also lead to overall improvements in performance similar to fixed over-correction. It may also be worthwhile to investigate whether a suitable fixed over-correction parameter can be efficiently determined a priori by considering a small test problem, or by analytic means.

7.2 Canonical tensor decomposition

7.2.1 Contributions and future research

In Chapter 6 we discussed how techniques from adaptive multigrid can be used to accelerate the alternating least squares method for computing the CP decomposition of higher-order tensors. As far as we are aware, our method is the first genuine multigrid algorithm for computing the CP decomposition as well as the first adaptive AMG method for solving a nonlinear optimization problem. Our method consists of two phases: a multiplicative correction scheme with bootstrap AMG interpolation as the setup phase, and an additive correction scheme based on the full approximation scheme as the solve phase. Numerical tests with dense and sparse tensors of varying sizes and orders (up to order eight) that are related to PDE problems demonstrated that our multilevel method can lead to significant speedup over standalone ALS when high accuracy is desired. Furthermore, a test case that is unrelated to PDE problems demonstrated that our multilevel method may be successfully applied to more general test problems.

The main area in which our algorithm can be improved is its robustness. The often erratic convergence behavior of the ALS method can make it difficult to construct sufficiently accurate transfer operators during the setup phase. Therefore, it may be beneficial to consider a Tikhonov-regularized formulation of CP [1]. As discussed in [120], the regularized problem is always well-posed in that it admits a global minimizer. Moreover, numerical investigation has shown that *swamps* in ALS convergence (regions of very slow error reduction) are less likely to occur with the regularized formulation. A more general coarsening routine would also lead to improved robustness in that a wider class of tensors could be considered. As in classical AMG, a rigorous (at the very least heuristic) characterization of the error for certain classes of tensors will be key in guiding the development of an automatic operator-based coarsening routine. It may also be worthwhile to investigate a more sophisticated setup phase that iteratively combines the CP-AMG-mult cycles and CP-FAS-FMG cycles, and to investigate more sophisticated measures of convergence stagnation. How best to deal with stagnation is yet another question. As discussed in [47], the addition of a line search on the finest level may also improve convergence and robustness. Furthermore, combining our AMG method with the GMRES method for canonical tensor decomposition developed in [46] may also prove fruitful.

Further avenues of research include identifying classes of tensors for which multigrid acceleration of ALS is beneficial, developing an alternative formulation of the coarse-level equations without the inverted Cholesky factors, and generalizing our multilevel framework to other similar tensor optimization problems such as the Tucker decomposition [79], block tensor decompositions [41, 44], and the best rank-$(R_1, \ldots, R_N)$ approximations [42, 70]. A class of tensors that we feel our method would be suitable for and that we are eager to experiment with in the future are derived from time series of still images that arise in applications such as medical imaging.

References

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *J. Chemometrics*, 25(2):67–86, 2011.

[2] R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 2:430–454, 1981.

[3] B. W. Bader and T. G. Kolda. Tensor Toolbox for Matlab, version 2.4. Available online http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/. Last accessed June, 2010.

[4] J. Ballani and L. Grasedyck. A projection method to solve linear systems in tensor format. *Numer. Linear Algebra Appl.*, 20(1):27–43, 2012.

[5] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1993.

[6] F. Bause and P. S. Kritzinger. *Stochastic Petri Nets*. Verlag, Berlin, Germany, 1996.

[7] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley, Hoboken, NJ, 3rd edition, 2006.

[8] M. Benzi and V. Kuhlemann. Restricted additive Schwarz methods for Markov chains. *Numer. Linear Algebra Appl.*, 00:1–20, 2000.

[9] M. Benzi and M. Tůma. A parallel solver for large-scale Markov chains. *Appl. Numer. Math.*, 41:135–153, 2002.

[10] M. Benzi and B. Uçar. Block triangular preconditioners for M-matrices and Markov chains. *Electron. Trans. Numer. Anal.*, 26:209–227, 2007.

[11] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[12] D. Bertsimas and J. N. Tsitisklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, MA, 1997.

[13] R. Blaheta. A multilevel method with overcorrection by aggregation for solving discrete elliptic problems. *J. Comput. Appl. Math.*, 24:227–239, 1988.

[14] R. G. Bland, D. Goldfarb, and M. J. Todd. The ellipsoid method: A survey. *Oper. Res.*, 29(6):1039–1091, 1981.

[15] M. Bolten, A. Brandt, J. Brannick, A. Frommer, K. Khal, and I. Livshits. A bootstrap algebraic multigrid method for Markov chains. *SIAM J. Sci. Comput.*, 33(6):3425–3446, 2011.

[16] S. Börm and R. Hiptmair. Analysis of tensor product multigrid. *Numer. Algorithms*, 26:219–234, 2001.

[17] D. Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55:379–393, 1995.

[18] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31(138):333–390, 1977.

[19] A. Brandt. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comput.*, 19:23–56, 1986.

[20] A. Brandt. Multiscale scientific computation: Review 2001. In T. Barth, T. Chan, and R. Haimes, editors, *Multiscale and Multiresolution Methods: Theory and Applications*. Springer, 2002.

[21] A. Brandt, J. Brannick, K. Kahl, and I. Livshits. Bootstrap AMG. *SIAM J. Sci. Comput.*, 33(2):612–632, 2011.

[22] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations. Technical report, Institute for Computational Studies, Colorado State University, Ft. Collins, Colorado, October 1982.

[23] A. Brandt, S. F. McCormick, and J. W. Ruge. Multilevel methods for differential eigenproblems. *SIAM J. Sci. Stat. Comput.*, 4(2):244–260, 1983.

[24] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and its Applications*. Cambridge University Press, 1984.

[25] A. Brandt and V. Mikulinsky. On recombining iterants in multigrid algorithms and problems with small islands. *SIAM J. Sci. Comput.*, 16:20–28, 1995.

[26] A. Brandt and D. Ron. Multigrid solvers and multilevel optimization strategies. In J. Cong and J. R. Shinnerl, editors, *Multilevel Optimization and VLSICAD*. Kluwer, 2003.

[27] A. Brandt and I. Yavneh. Accelerated multigrid convergence and high-Reynolds recirculating flows. *SIAM J. Sci. Comput.*, 14(3):607–626, 1993.

[28] J. Brannick and R. D. Falgout. Compatible relaxation and coarsening in algebraic multigrid. *SIAM J. Sci. Comput.*, 32:1393–1416, 2009.

[29] M. Brezina, R. D. Falgout, S. MacLachlan, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge. Adaptive smoothed aggregation (αSA) multigrid. *SIAM Rev. SIGEST*, 47:317–346, 2004.

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

[31] M. Brezina, T. A. Manteuffel, S. F. McCormick, J. W. Ruge, and G. Sanders. Towards adaptive smooth aggregation (αSA) for nonsymmetric problems. *SIAM J. Sci. Comput.*, 32:14–39, 2010.

[32] W. L. Briggs, V. Henson, and S. F. McCormick. *A Multigrid Tutorial.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2nd edition, 2000.

[33] P. Buchholz. Multilevel solutions for structured Markov chains. *SIAM J. Matrix Anal. Appl.*, 22(2):342–357, 2000.

[34] W.-L. Cao and W. J. Stewart. Iterative aggregation/disaggregation techniques for nearly uncoupled Markov chains. *J. ACM*, 32(3):702–719, 1985.

[35] J. D. Carroll and J. J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35:283–319, 1970.

[36] G. Chartrand and L. Lesniak. *Graphs & Digraphs.* Chapman and Hall/CRC, Boca Raton, FL, 4th edition, 2005.

[37] F. Chatelin and W. L. Miranker. Acceleration by aggregation of successive approximation methods. *Linear Algebra Appl.*, 43:17–47, 1982.

[38] A. J. Cleary, R. D. Falgout, V. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick G. N. Miranda, and J. W. Ruge. Robustness and scalability of algebraic multigrid. *SIAM J. Sci. Comput.*, 21(5):1886–1908, December 1999.

[39] Tuğrul Dayar and William J. Stewart. Comparison of partitioning techniques for two-level iterative solvers on large, sparse, Markov chains. *SIAM J. Sci. Comput.*, 21:1691–1705, 2000.

[40] L. De Lathauwer. A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. *SIAM J. Matrix Anal. Appl.*, 28:642–666, 2006.

[41] L. De Lathauwer. Decompositions of a higher-order tensor in block terms—part II: Definitions and uniqueness. *SIAM J. Matrix Anal. Appl.*, 30(3):1033–1066, 2008.

[42] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, \ldots, R_N)$ approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.

[43] L. De Lathauwer, B. De Moor, and J. Vandewalle. Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition. *SIAM J. Matrix Anal. Appl.*, 26:295–327, 2004.

[44] L. De Lathauwer and D. Nion. Decompositions of a higher-order tensor in block terms—part III: Alternating least squares algorithms. *SIAM J. Matrix Anal. Appl.*, 30(3):1067–1083, 2008.

[45] V. de Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM. J. Matrix Anal. Appl.*, 30:1084–1127, 2008.

[46] H. De Sterck. A nonlinear GMRES optimization algorithm for canonical tensor decomposition. *SIAM J. Sci. Comput.*, 34(3):A1351–A1379, 2012.

[47] H. De Sterck. A self-learning algebraic multigrid method for extremal singular triplets and eigenpairs. *SIAM J. Sci. Comput.*, 34(4):A2092–A2117, 2012.

[48] H. De Sterck, J. J. Heys, and U. M. Yang. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM J. Matrix Anal. Appl.*, 27(4):1019–1039, 2006.

[49] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Pearson, J. W. Ruge, and G. Sanders. Smoothed aggregation multigrid for markov chains. *SIAM J. Sci. Comput.*, 32:40–61, 2010.

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

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

[52] H. De Sterck, T. A. Manteuffel, K. Miller, and G. Sanders. Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains. *Adv. Comput. Math.*, 35:375–403, 2010.

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

[54] S. T. Dziuban, J. G. Ecker, and M. Kupferschmid. Using deep cuts in an ellipsoid algorithm for nonlinear programming. *Math. Program. Stud.*, 25:93–107, 1985.

[55] J. G. Ecker and M. Kupferschmid. An ellipsoid algorithm for nonlinear programming. *Math. Program.*, 27:83–106, 1983.

[56] R. D. Falgout. An introduction to algebraic multigrid. Technical Report UCRL-JRNL-220851, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, California, April 2006.

[57] J. B. G. Frenk, J. Gromicho, and S. Zhang. A deep cut ellipsoid algorithm for convex programming: Theory and applications. *Math. Program.*, 63:83–108, 1994.

[58] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, 28(126):505–535, 1974.

[59] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Trans. Math. Software*, 10(3):282–298, 1984.

[60] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[61] W. Grassmann, M. Taksar, and D. Heyman. Regenerative analysis and steady-state distributions for Markov chains. *Oper. Res.*, 33(5):1107–1116, 1985.

[62] H. Guillard and P. Vaněk. An aggregation multigrid solver for convection-diffusion problems on unstructured meshes. Technical Report UCD-CCM-130, Center for Computational Mathematics, University of Colorado at Denver, May 1998.

[63] H.-J. Lüthi. On the solution of variational inequalities by the ellipsoid method. *Math. Oper. Res.*, 10(3):515–522, 1985.

[64] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[65] M. Haviv. Aggregation/disaggregation methods for computing the stationary distribution of Markov chains. *SIAM J. Numer. Anal.*, 24(4):952–966, 1987.

[66] V. Henson and P. S. Vassilevski. Element-free AMGe: General algorithms for computing interpolation weights in AMG. *SIAM J. Sci. Comput.*, 23(2):629–650, 2001.

[67] C. J. Hillar and L.-H. Lim. Most tensor problems are NP-hard. Preprint, arXiv:0911.1393v3 [cs.CC], 2012.

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

[69] G. Horton and S. T. Leutenegger. A multi-level solution algorithm for steady-state Markov chains. *ACM SIGMETRICS Perform. Eval. Rev.*, 22(1):191–200, 1994.

[70] M. Ishteva. *Numerical methods for the best low multilinear rank approximation of higher-order tensors*. PhD thesis, K. U. Leuven, 2009.

[71] D. B. Iudin and A. S. Nemirovskii. Informational complexity and effective methods of solution for convex extremal problems. *Matekon: Translations of Russian and East European Math. Economics*, 13:3–25, 1976.

[72] K. Kahl. *Adaptive Algebraic Multigrid for Lattice QCD Computations*. PhD thesis, University of Wuppertal, 2009.

[73] H. B. Keller. On the solution of singular and semidefinite linear systems by iteration. *J. SIAM Numer. Anal. Ser. B*, 2(2):281–290, 1965.

[74] L. G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1976.

[75] B. Khoromskij. On tensor approximation of Green iterations for Kohn–Sham equations. *Comput. Visual. Sci.*, 11:259–271, 2008.

[76] B. Khoromskij. Tensor-structured preconditioners and approximate inverse of elliptic operators in $\mathbb{R}^d$. *Constr. Approx.*, 30:599–620, 2009.

[77] B. N. Khoromskij and V. Khoromskaia. Multigrid accelerated tensor approximation of function related multidimensional arrays. *SIAM J. Sci. Comput.*, 31(4):3002–3026, 2009.

[78] T. G. Kolda. Multilinear operators for higher-order decompositions. Technical Report SAND2006-2081, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, CA, 2006.

[79] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.

[80] J. R. Koury, D. F. McAllister, and W. J. Stewart. Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains. *SIAM J. Alg. Discr. Meth.*, 5(2):164–186, 1984.

[81] U. R. Krieger. On a two-level multigrid solution method for Markov chains. *Linear Algebra Appl.*, 223–224:415–438, 1995.

[82] U. R. Krieger, B. Müller-Clostermann, and M. Sczittnick. Modeling and analysis of communication systems based on computational methods for Markov chains. *IEEE J. Selected Areas Commun.*, 8(9):1630–1648, 1990.

[83] D. Kushnir, M. Galun, and A. Brandt. Efficient multilevel eigensolvers with applications to data analysis tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1377–1391, 2010.

[84] W. Leontief. *The Structure of the American Economy 1919–1939.* Oxford University Press, New York, NY, 1951.

[85] S. T. Leutenegger and G. Horton. On the utility of the multi-level algorithm for the solution of nearly completely decomposable Markov chains. In *Second International Workshop on the Numerical Solution of Markov Chains.* Kluwer, 1995.

[86] I. Livshits. An algebraic multigrid wave-ray algorithm to solve eigenvalue problems for the Helmholtz operator. *Numer. Linear Algebra Appl.*, 11:229–239, 2004.

[87] J. Mandel and B. Sekerka. A local convergence proof for the iterative aggregation method. *Linear Algebra Appl.*, 51:163–172, 1983.

[88] I. Marek and P. Mayer. Convergence analysis of an iterative aggregation/disaggregation method for computing stationary probability vectors of stochastic matrices. *Numer. Linear Algebra Appl.*, 5:253–274, 1998.

[89] I. Marek and P. Mayer. Convergence theory of some classes of iterative aggregation/disaggregation methods for computing stationary probability vectors of stochastic matrices. *Linear Algebra Appl.*, 363:177–200, 2003.

[90] I. Marek and D. B. Szyld. Algebraic Schwarz methods for the numerical solution of Markov chains. *Linear Algebra Appl.*, 386:67–81, 2004.

[91] C. D. Meyer. Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. *SIAM Rev.*, 31(2):240–272, 1989.

[92] M. Molloy. Performance analysis using stochastic Petri nets. *IEEE Trans. Comput.*, C-31(9):913–917, 1982.

[93] M. Neumann and R. J. Plemmons. Convergent nonnegative matrices and iterative methods for consistent linear systems. *Numer. Math.*, 31:265–279, 1978.

[94] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, NY, 2nd edition, 2006.

[95] J. R. Norris. *Markov Chains.* Cambridge University Press, New York, NY, 1997.

[96] Y. Notay. Algebraic analysis of two-grid methods: The nonsymmetric case. *Numer. Linear Algebra Appl.*, 17:73–96, 2010.

[97] I. V. Oseledets and D. V. Savost'yanov. Minimization methods for approximating tensors and their comparison. *Comp. Math. Math. Phys.*, 46(10):1641–1650, 2006.

[98] I. V. Oseledets, D. V. Savost'yanov, and E. E. Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM J. Matrix Anal. Appl.*, 30(3):939–956, 2008.

[99] P. Buchholz and T. Dayar. Comparison of multilevel methods for Kronecker-based Markovian representations. *Computing*, 73:349–371, 2004.

[100] B. Philippe, Y. Saad, and W. J. Stewart. Numerical methods for Markov chain modeling. *Oper. Res.*, 40:1156–1179, 1992.

[101] M. Rajih, P. Comon, and R. A. Harshman. Enhanced line search: A novel method to accelerate PARAFAC. *SIAM J. Matrix Anal. Appl.*, 30(3):1128–1147, 2008.

[102] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, NJ, 1970.

[103] J. W. Ruge and K. Stüben. Algebraic multigrid. In S. F. McCormick, editor, *Multigrid Methods, Frontiers in Applied Mathematics.* Society for Industrial and Applied Mathematics, 1987.

[104] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2nd edition, 2003.

[105] B. Seibold. Performance of algebraic multigrid methods for non-symmetric matrices arising in particle methods. *Numer. Linear Algebra Appl.*, 17:433–451, 2010.

[106] N. Z. Shor. Cut-off method with space extension in convex programming problems. *Cybernetics*, 13:94–96, 1977.

[107] H. A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29:111–138, 1961.

[108] R. V. Southwell. *Relaxation Methods in Theoretical Physics.* Clarendon Press, Oxford, 1946.

[109] W. J. Stewart. MARCA Models: A Collection of Markov Chain Models. Available online http://www4.ncsu.edu/~billy/MARCA_Models/MARCA_Models.html. Last accessed August 2008.

[110] W. J. Stewart. *An Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, Princeton, NJ, 1994.

[111] W. J. Stewart. *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling.* Princeton University Press, Princeton, NJ, 2009.

[112] K. Stüben. Algebraic multigrid (AMG): An introduction with applications. Technical Report GMD Report 70, GMD - Forschungszentrum Informationstechnik GmbH, Sankt Augustin, Germany, November 1999.

[113] G. Styan. Hadamard products and multivariate statistical analysis. *Linear Algebra Appl.*, 6:217–240, 1973.

[114] Y. Takahashi. A lumping method for numerical calculations of stationary distributions of Markov chains. Technical Report B-18, Department of Information Sciences, Tokyo Institute of Technology, June 1975.

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

[116] G. Tomasi and R. Bro. A comparison of algorithms for fitting the PARAFAC model. *Comput. Stat. Data Anal.*, 50:1700–1734, 2006.

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

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

[119] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Elsevier Academic Press, San Diego, CA, 2001.

[120] A. Uschmajew. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM J. Matrix Anal.*, 33(2):639–652, 2012.

[121] P. Vaněk. Acceleration of convergence of a two-level algorithm by smoothing transfer operators. *Appl. Math.*, 37(4):265–274, 1992.

[122] P. Vaněk. Fast multigrid solver. *Appl. Math.*, 40(1):1–20, 1995.

[123] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid on unstructured meshes. Technical Report UCD-CCM-034, Center for Computational Mathematics, University of Colorado at Denver, December 1994.

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

[125] P. Vaněk and S. Míka. Modification of two-level algorithm with overcorrection. *Appl. Math.*, 37:13–28, 1992.

[126] R. S. Varga. *Matrix Iterative Analysis*. Springer, Berlin, 2nd edition, 2000.

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

[128] T. Washio and C. W. Oosterlee. Krylov subspace acceleration for nonlinear multigrid schemes. *Electron. Trans. Numer. Anal.*, 6:271–290, 1997.

[129] J. Zhang. Minimal residual smoothing in multi-level iterative method. *Appl. Math. Comput.*, 84:1–25, 1997.

[130] J. Zhang. Residual scaling techniques in multigrid, I: Equivalence proof. *Appl. Math. Comput.*, 86:283–303, 1997.

[131] J. Zhang. Residual scaling techniques in multigrid, II: Practical applications. *Appl. Math. Comput.*, 90:229–252, 1998.