submitted to SIAM Journal on Scientific Computing November 2011; revised version submitted May 2012

AN ADAPTIVE ALGEBRAIC MULTIGRID ALGORITHM FOR LOW-RANK CANONICAL TENSOR DECOMPOSITION*

HANS DE STERCK† AND KILLIAN MILLER†

Abstract. A new algorithm based on algebraic multigrid is presented for computing the rank-R canonical decomposition of a tensor for small R. Standard alternating least squares (ALS) is used as the relaxation method. Transfer operators and coarse-level tensors are constructed in an adaptive setup phase that combines multiplicative correction and bootstrap algebraic multigrid. An accurate solution is computed by an additive solve phase based on the Full Approximation Scheme. Numerical tests show that for certain test problems our multilevel method significantly outperforms standalone ALS when a high level of accuracy is required.

Key words. multilevel method, algebraic multigrid, numerical multilinear algebra, canonical tensor decomposition, CANDECOMP, PARAFAC, alternating least squares

AMS subject classifications. 65N55 Multigrid methods, 65F10 Iterative methods, 15A69 Multilinear algebra

1. Introduction. This paper presents a multigrid method for accurately computing a low-rank canonical decomposition of a tensor. An Nth-order tensor is an N-dimensional array of size $I_1 \times \cdots \times I_N$ [23]. The *order* of a tensor is the number of modes (dimensions), and the size of the nth mode is I_n for $n = 1, \ldots, N$. 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 $\mathcal{T}$ is an Nth-order tensor of rank R, which implies that $\mathcal{T}$ can be expressed as a sum of no fewer than R rank-one components, then its canonical decomposition is given by

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \cdots \circ \mathbf{a}_r^{(N)}, \tag{1.1}$$

where $\circ$ denotes the vector outer product. The rth rank-one component in (1.1) 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 $n = 1, \ldots, N$, one can store the vectors $\mathbf{a}_1^{(n)}, \ldots, \mathbf{a}_R^{(n)}$ as the columns of an $I_n \times R$ matrix $\mathbf{A}^{(n)}$. The matrix $\mathbf{A}^{(n)}$ is referred to as the mode-n *factor matrix* and its columns are the mode-n *factors*. The canonical tensor decomposition may then be expressed in terms of the factor matrices by the double bracket notation $[\![\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!]$, which is defined as the summation in (1.1).

We refer to the canonical decomposition as CANDECOMP/PARAFAC (CP) after the names originally given to it in early papers on the subject [14, 15]. Whereas (1.1) is an exact decomposition, our aim is to find the "best" approximate decomposition (in some sense) for an arbitrary Nth-order tensor $\mathcal{Z}$ and a given number of components R. The problem of computing the CP decomposition that best approximates $\mathcal{Z}$ may

*This work was supported by NSERC of Canada and was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

†Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada (hdesterck@uwaterloo.ca, k7miller@uwaterloo.ca).

be formulated as a least squares optimization problem:

$$\text{minimize} \quad f(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}) \coloneqq \frac{1}{2}\left\|\mathcal{Z} - [\![\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}]\!]\right\|^2. \tag{1.2}$$

Here, $\|\cdot\|$ denotes the Frobenius norm of a tensor, defined as the square root of the sum of the squared entries of all tensor elements. In what follows, for matrices $\|\cdot\|$ refers to the Frobenius norm, and for vectors it refers to the vector two-norm. A general approach to solving (1.2) is to satisfy the *first-order optimality equations* (see §3), i.e., to find a set of nontrivial factor matrices that zero out the gradient of f. In recent work on computing CP, Acar, Dunlavy and Kolda [1] consider standard gradient-based optimization methods such as the nonlinear conjugate gradient method for optimization problem (1.2), and compare their approach with alternating least squares (ALS) and nonlinear least squares algorithms. Similarly, De Sterck [34] proposes a new method for computing CP that uses a nonlinear generalized minimal residual method to solve the optimality equations directly. In this paper, we compute (local) minimizers of f by using algebraic multigrid (AMG) techniques to solve the optimality equations.

Optimization problem (1.2) is non-convex, consequently it may admit multiple local minima. Moreover, for any local minimizer there is a continuous manifold of equivalent minimizers [23]. This manifold arises because of a scaling indeterminacy inherent to the CP decomposition, i.e., 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 [23]. In the following sections we explain how the scaling and permutation indeterminacies can be removed by imposing a specific normalization and ordering of the factors. However, in spite of normalizing and ordering the factors the CP decomposition may still exhibit multiple local minima for some tensors, and depending on the initial guess, iterative methods may converge to different stationary points. Furthermore, for certain tensors and certain values of R a best rank-R approximation does not exist [33]. 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 decompositions are used in many application fields [23].

The primary application of the CP decomposition is as a tool for data analysis, where it has been used in a variety of fields including chemometrics, data mining, image compression, neuroscience and telecommunications. A second class of problems is related to the decomposition of tensors arising from PDE discretization on high-dimensional regular lattices [19, 20, 30, 31]. Many algorithms have been proposed for computing the CP decomposition [1, 23, 25, 28, 38], however, the workhorse algorithm is the original alternating least squares (ALS) method, first proposed in 1970 in early papers on the CP decomposition [14, 15]. The alternating least squares method is simple to implement, and often performs adequately, however, it can be very 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 upon ALS in a robust way for large classes of problems. Accordingly, ALS-type algorithms remain the method of choice in practice.

The multigrid method described in this paper consists of two multilevel phases: a multiplicative correction scheme as the setup phase, and an additive correction scheme as the solve phase. In the setup phase a multiplicative correction scheme is

used in conjunction with bootstrap algebraic multigrid (BAMG) interpolation [8, 17] to not only build the necessary transfer operators and coarse-level tensors, but also to compute initial approximations of the factor matrices. In this phase the 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, since 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 [24], where it formed the basis of an efficient eigensolver for multiclass spectral clustering problems. A similar approach was also proposed in [9, 17]. In the solve phase we use the Full Approximation Scheme (FAS) [7, 10, 13, 40] to efficiently obtain an accurate solution. Our multigrid framework is closely related to recent work on an adaptive algebraic multigrid solver for extremal singular triplets and eigenpairs of matrices [35], and to a lesser degree to multigrid methods for Markov chains [4, 36, 39]. We note that while our proposed algorithm is the first such multigrid method for computing the CP decomposition of a tensor, the idea to apply multilevel methods to problems in multilinear algebra has already been discussed and analyzed, for example, see [3, 5, 21]. In this paper we provide a complete but *concise* description of the proposed algorithm; readers who are unfamiliar with adaptive algebraic multigrid (AMG) methods can find background information about the general adaptive AMG approach in [4, 9, 11, 12, 35, 39].

We expect our method to work well for tensors that have properties which 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 [19, 20, 30, 31], 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 in this paper 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 [35].

The remainder of this paper is structured as follows. Section 2 presents the notation and basic operations used throughout this paper. Section 3 presents the first-order optimality equations and describes the alternating least squares method. Section 4 describes the multilevel setup phase, and §5 describes the multilevel solve phase. Implementation details and numerical results are presented in §6 followed by

concluding remarks in §7.

2. Notation. This section presents our notation, much of which has been adopted from [1]. Basic operations and identities that are important to this paper are also provided. For further details we refer to the survey article by Kolda and Bader [23], and the extensive references therein.

Vectors (tensors of order one) are denoted by boldface lowercase letters, e.g., $\mathbf{v}$. Matrices (tensors of order two) are denoted by boldface capital letters, e.g., $\mathbf{A}$. Higher-order tensors are denoted by boldface Euler script letters, e.g., $\mathcal{Z}$. The ith entry of a vector $\mathbf{v}$ is denoted by v_i, element (i,j) of a matrix $\mathbf{A}$ is denoted by a_{ij}, and, for example, element (i,j,k,ℓ) of a fourth-order tensor $\mathcal{Z}$ is denoted by $z_{ijk\ell}$. The jth column of a matrix $\mathbf{A}$ is denoted by $\mathbf{a}_j$. The nth element of a sequence is denoted by a superscript in parentheses, e.g., $\mathbf{A}^{(n)}$. In general, indices range from 1 to their capital versions, e.g., $n = 1, \ldots, N$.

Matricization, also referred to as unfolding or flattening, is the process of reordering the elements of a tensor into a matrix. In this paper we are only interested in mode-n matricization, which arranges the mode-n *fibers* to be the columns of the resulting matrix. Note that a fiber is a higher-order analogue of matrix rows/columns, which is obtained by fixing every index of a tensor but one. 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

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

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

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

where $\otimes$ is the *Kronecker product*. Associativity of the Khatri-Rao product and the *mixed-product property* of the Kronecker product, i.e., $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$, imply the following useful result:

$$\mathbf{A}^{(1)}\mathbf{B}^{(1)} \odot \cdots \odot \mathbf{A}^{(N)}\mathbf{B}^{(N)} = \left(\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)}\right)\left(\mathbf{B}^{(1)} \odot \cdots \odot \mathbf{B}^{(N)}\right) \tag{2.1}$$

for any sequences of matrices $\mathbf{A}^{(n)}$ and $\mathbf{B}^{(n)}$, for $n = 1, \ldots, N$, of the appropriate sizes. An important relationship between tensors and their matricized versions is as follows. Let $\mathcal{Z} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\mathbf{A}^{(n)} \in \mathbb{R}^{J_n \times I_n}$ for $n = 1, \ldots, N$. Then, for any $n \in \{1, \ldots, N\}$

$$\mathcal{X} = \mathcal{Z} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)} \quad \Leftrightarrow \quad \mathbf{X}_{(n)} = \mathbf{A}^{(n)}\mathbf{Z}_{(n)}\left(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(1)}\right)^T. \tag{2.2}$$

A proof of this property is given in [22].

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

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

3. CP first-order optimality equations and alternating least squares. The first-order optimality equations for the CP decomposition are obtained by setting the gradient of the functional in (1.2) equal to zero. Following the derivation in [1], for each mode $n \in \{1, \dots, N\}$ the derivative of f with respect to $\mathbf{A}^{(n)}$ can be written as an $I_n \times R$ matrix

$$\mathbf{G}^{(n)} = -\mathbf{Z}_{(n)}\mathbf{\Phi}^{(n)} + \mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)}, \tag{3.1}$$

where

$$\mathbf{\Phi}^{(n)} = \mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}, \tag{3.2}$$

and

$$\mathbf{\Gamma}^{(n)} = \mathbf{\Upsilon}^{(1)} * \cdots * \mathbf{\Upsilon}^{(n-1)} * \mathbf{\Upsilon}^{(n+1)} * \cdots * \mathbf{\Upsilon}^{(N)} \tag{3.3}$$

with $\mathbf{\Upsilon}^{(n)} = {\mathbf{A}^{(n)}}^T\mathbf{A}^{(n)}$ for $n = 1, \dots, N$. The first-order optimality equations are then given by

$$\mathbf{G}^{(n)} = \mathbf{0}, \quad n = 1, \dots, N. \tag{3.4}$$

We note that since the $R \times R$ matrix $\mathbf{\Gamma}^{(n)}$ is the Hadamard (element-wise) product $*$ of symmetric positive semidefinite matrices, it too must be symmetric positive semidefinite [37]. Moreover, if each $\mathbf{A}^{(n)}$ has full rank then $\mathbf{\Gamma}^{(n)}$ will be symmetric positive definite (SPD).

One iteration of ALS for the CP decomposition is equivalent to applying one iteration of block nonlinear Gauss–Seidel (BNGS) to the optimality equations (3.4). 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 $\mathbf{A}^{(n)}$. In particular, $\mathbf{\Gamma}^{(n)}$ and $\mathbf{\Phi}^{(n)}$ are updated and $\mathbf{A}^{(n)} \leftarrow \mathbf{Z}_{(n)}\mathbf{\Phi}^{(n)}(\mathbf{\Gamma}^{(n)})^\dagger$, where $(\mathbf{\Gamma}^{(n)})^\dagger$ is the Moore–Penrose pseudoinverse of $\mathbf{\Gamma}^{(n)}$. Owing to the scaling indeterminacy inherent to the CP decomposition, it is possible that 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 complete ALS iteration, the factors of the rth component are normalized according to

$$\mathbf{a}_r^{(n)} \mapsto \lambda_r \left(\frac{\mathbf{a}_r^{(n)}}{\|\mathbf{a}_r^{(n)}\|} \right) \quad \text{for } n = 1, \dots, N, \quad \lambda_r = \left(\|\mathbf{a}_r^{(1)}\| \dots \|\mathbf{a}_r^{(N)}\| \right)^{1/N} \tag{3.5}$$

for $r = 1, \dots, R$. This normalization equilibrates the norms of the factors of each component, i.e., $\|\mathbf{a}_r^{(1)}\| = \cdots = \|\mathbf{a}_r^{(N)}\|$ for $r = 1, \dots, R$. The ALS algorithm described here is used as the relaxation method and coarsest-level solver in the setup phase. We note that upon completion of the ALS iterations the rank-one terms are sorted in decreasing order of the normalization factors λ_r.

4. 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. Our multiplicative setup phase uses the bootstrap AMG approach described in [4, 8, 9, 17, 24, 35]. Moreover, our setup phase is similar to the setup phase of [35] for computing SVD triplets of a matrix, in the sense that it uses a separate interpolation matrix for each mode.

4.1. Derivation of coarse-level equations. The fine-level equations correspond to the first-order optimality equations (see §3) given by

$$\mathbf{Z}_{(n)}\Big(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}\Big) = \mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)} \tag{4.1}$$

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, for each n, $\mathbf{A}^{(n)} \approx \mathbf{P}^{(n)}\mathbf{A}_c^{(n)}$ for some coarse-level variable $\mathbf{A}_c^{(n)} \in \mathbb{R}^{I_{n,c} \times R}$ (since each factor matrix has R columns it is unlikely that equality can be achieved). Then a solution of (4.1) can be approximated by solving a coarse-level problem

$$\begin{aligned}&{\mathbf{P}^{(n)}}^T\mathbf{Z}_{(n)}\Big(\mathbf{P}^{(N)}\mathbf{A}_c^{(N)} \odot \cdots \odot \mathbf{P}^{(n+1)}\mathbf{A}_c^{(n+1)} \odot \mathbf{P}^{(n-1)}\mathbf{A}_c^{(n-1)} \odot \cdots \odot \mathbf{P}^{(1)}\mathbf{A}_c^{(1)}\Big)\\ &= \Big({\mathbf{P}^{(n)}}^T\mathbf{P}^{(n)}\Big)\mathbf{A}_c^{(n)}\mathbf{\Gamma}_c^{(n)} \quad \text{for } n = 1, \ldots, N,\end{aligned} \tag{4.2}$$

followed by interpolation. Here, $\mathbf{\Gamma}_c^{(n)}$ is defined as in (3.3) with

$$\mathbf{\Upsilon}_c^{(n)} = {\mathbf{A}_c^{(n)}}^T\Big({\mathbf{P}^{(n)}}^T\mathbf{P}^{(n)}\Big)\mathbf{A}_c^{(n)} \quad \text{for } n = 1, \ldots, N.$$

Letting $\mathbf{B}^{(n)} = {\mathbf{P}^{(n)}}^T\mathbf{P}^{(n)}$ for each mode n, and using properties (2.2) and (2.1), the coarse-level problem (4.2) can be written as

$$\mathbf{Z}_{(n)}^c\Big(\mathbf{A}_c^{(N)} \odot \cdots \odot \mathbf{A}_c^{(n+1)} \odot \mathbf{A}_c^{(n-1)} \odot \cdots \odot \mathbf{A}_c^{(1)}\Big) = \mathbf{B}^{(n)}\mathbf{A}_c^{(n)}\mathbf{\Gamma}_c^{(n)}, \tag{4.3}$$

where the coarse-level tensor is given by

$$\mathcal{Z}^c = \mathcal{Z} \times_1 {\mathbf{P}^{(1)}}^T \times_2 {\mathbf{P}^{(2)}}^T \cdots \times_N {\mathbf{P}^{(N)}}^T. \tag{4.4}$$

Note that (4.4) 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 (4.3), 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 $\hat{\mathbf{A}}_c^{(n)} = {\mathbf{L}^{(n)}}^T\mathbf{A}_c^{(n)}$ for $n = 1, \ldots, N$, and again appealing to properties (2.2) and (2.1), it follows that (4.3) can be written as

$$\hat{\mathbf{Z}}_{(n)}^c\Big(\hat{\mathbf{A}}_c^{(N)} \odot \cdots \odot \hat{\mathbf{A}}_c^{(n+1)} \odot \hat{\mathbf{A}}_c^{(n-1)} \odot \cdots \odot \hat{\mathbf{A}}_c^{(1)}\Big) = \hat{\mathbf{A}}_c^{(n)}\hat{\mathbf{\Gamma}}_c^{(n)},$$

where the transformed coarse-level tensor is given by

$$\hat{\mathcal{Z}}^c = \mathcal{Z} \times_1 \hat{\mathbf{P}}^{(1)^T} \times_2 \hat{\mathbf{P}}^{(2)^T} \cdots \times_N \hat{\mathbf{P}}^{(N)^T}, \tag{4.5}$$

with $\hat{\mathbf{P}}^{(n)} = \mathbf{P}^{(n)}{\mathbf{L}^{(n)}}^{-T}$ 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:

$$\hat{f}_c(\hat{\mathbf{A}}_c^{(1)}, \ldots, \hat{\mathbf{A}}_c^{(N)}) \coloneqq \frac{1}{2}\Big\|\hat{\mathcal{Z}}^c - [\![\hat{\mathbf{A}}_c^{(1)}, \ldots, \hat{\mathbf{A}}_c^{(N)}]\!]\Big\|^2. \tag{4.6}$$

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:

$$\mathbf{A}_{\text{CGC}}^{(n)} = \hat{\mathbf{P}}^{(n)}\hat{\mathbf{A}}_c^{(n)} \quad \text{for } n = 1, \ldots, N. \tag{4.7}$$

Now suppose that $\mathbf{P}^{(n)}$ contains $\mathbf{A}^{(n)}$ exactly in its range, and hence that $\hat{\mathbf{R}}^{(n)}\mathbf{A}^{(n)} = \hat{\mathbf{A}}_c^{(n)}$. Further suppose that $\hat{\mathbf{A}}_c^{(n)}$ is a fixed point of the coarse-level solver. Then

$$\mathbf{A}_{\text{CGC}}^{(n)} = \hat{\mathbf{P}}^{(n)}\hat{\mathbf{A}}_c^{(n)} = (\hat{\mathbf{P}}^{(n)}\hat{\mathbf{R}}^{(n)})\mathbf{A}^{(n)} = \mathbf{A}^{(n)} \quad \text{for } n = 1, \ldots, N.$$

However, since these assumptions are satisfied only approximately, we expect (4.7) to yield an improved but not exact approximation to the fine-level solution. In particular, since 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.

4.2. Bootstrap AMG V-cycles. We use bootstrap AMG to find initial approximations of the desired factor matrices, and to adaptively determine the interpolation operators that approximately fit the factor matrices.

We begin by describing the initial BAMG V-cycle. On the finest level we start with n_t *test blocks*, 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, since the rank-one components of the best rank-R CP tensor approximation must be found simultaneously [23]; 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 §3. 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 $\hat{\mathcal{Z}}^c$ 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 correction (CGC) is then relaxed by a few iterations of ALS. This process continues until the CGC on the finest level has been relaxed by ALS.

The initial BAMG V-cycle can be followed by several additional BAMG V-cycles. 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 for the additive phase of the algorithm, they must be well represented by interpolation if the additive solve phase is to converge.

4.3. 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 selecting a subset of the fine-level indices $\Omega_n = \{1, \ldots, I_n\}$. This subset, denoted by $\mathcal{C}_n$, is the set of coarse indices for mode n (it has cardinality $I_{n,c} < I_n$). Fine-level points that are not selected to the coarse level are represented by the set of fine indices $\mathcal{F}_n = \Omega_n \setminus \mathcal{C}_n$. 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

$$p_{ij}^{(n)} = \begin{cases} w_{ij}^{(n)}, & i \in \mathcal{F}_n \text{ and } j \in \mathcal{C}_n^i \\ 1, & i \in \mathcal{C}_n \text{ and } j = \alpha(i) \\ 0, & \text{otherwise,} \end{cases}$$

where the $w_{ij}^{(n)}$s are the interpolation weights for mode n. The interpolation weights are determined by a least-squares process described in §4.4. In this paper we use standard geometric coarsening for 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, for tensors in which the sizes of some modes vary widely, a more aggressive coarsening for the larger modes may be considered. In §4.5 we describe a straightforward approach to coarsening tensors with varying mode sizes. While the simple coarsening procedure discussed here works well for the test problems considered in §6 (PDE problems on high-dimensional regular lattices), more general coarsening algorithms for other types of tensors are desirable. The development of such algorithms remains an interesting topic of future research.

4.4. 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 test factor matrices 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 boot factor matrices. Following the approaches of [7, 17, 24, 35] we use a least-squares process 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 $\mathbf{U}_f$. 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 in the coarse-level point j. Also, let u_{ik} be the value of $\mathbf{u}_k$ in the fine-level point i. 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 $w_{ij}^{(n)}$,

$$u_{ik} = \sum_{j \in \mathcal{C}_n^i} w_{ij}^{(n)} (\mathbf{u}_{k,c})_j \quad \text{for } k = 1, \ldots, n_f. \tag{4.8}$$

We make (4.8) over-determined in all cases 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, $M_s = 2$, and 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 (4.8) 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

$$\mu_r = \frac{\|\mathbf{A}^{(n)}\|^2}{\|\mathbf{G}^{(n)}\|^2} \quad \text{for } r = 1, \ldots, R, \tag{4.9}$$

where $\mathbf{G}^{(n)}$ is the gradient in (3.1). 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 (4.9) 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 (4.9) since we do not want preferential treatment given to different factor vectors, but rather to entire factor matrices. In our implementation, the small weighted least-squares problems with weight matrix $\mathbf{M} = \operatorname{diag}(\boldsymbol{\mu})$ are solved via QR factorization.

4.5. Pseudocode. A pseudocode description for a multiplicative setup phase V-cycle with ALS as the relaxation scheme and coarsest-level solver is given by Algorithm 1. We note that a multigrid V-cycle is obtained by performing one recursive solve on each level. If two recursive solves are performed on each level then a multigrid W-cycle is obtained. In this paper we only consider V-cycles for the setup phase.

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 (4.5) 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

$$\tilde{\mathbf{A}}_c^{(n)} = \begin{cases} \hat{\mathbf{R}}^{(n)} \mathbf{A}^{(n)}, & n \in \mathcal{I}_c \\ \mathbf{A}^{(n)}, & n \in \mathcal{I}_c' \end{cases} \quad \text{for } n = 1, \ldots, N. \tag{4.10}$$

Algorithm 1: V-cycle for setup phase of CP decomposition (CP-AMG-mult)

Input: tensor $\mathcal{Z}$, BFMs $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$, TFMs
Output: updated BFMs $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$, updated TFMs

Compute the set $\mathcal{I}_c = \{n : I_n > I_{n,coarsest}\}$
if $\mathcal{I}_c \neq \emptyset$ **then**
  Apply ν_1 relaxations to TFMs in each test block and to $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$
  for $n \in \mathcal{I}_c$ **do**
    Build the interpolation operator $\mathbf{P}^{(n)}$ (on first cycle only use TFMs)
    Let $\mathbf{B}^{(n)} \leftarrow {\mathbf{P}^{(n)}}^T \mathbf{P}^{(n)}$
    Compute the Cholesky factorization $\mathbf{B}^{(n)} = \mathbf{L}^{(n)} {\mathbf{L}^{(n)}}^T$
    Let $\hat{\mathbf{P}}^{(n)} \leftarrow \mathbf{P}^{(n)} {\mathbf{L}^{(n)}}^{-T}$ and $\hat{\mathbf{R}}^{(n)} \leftarrow {\hat{\mathbf{P}}^{(n)}}^T$
  end
  Compute the coarse BFMs and coarse TFMs according to (4.10)
  Compute the coarse-level tensor $\hat{\mathcal{Z}}^c \leftarrow \mathcal{Z} \times_{n \in \mathcal{I}_c} \hat{\mathbf{R}}^{(n)}$
  Recursive solve:

$$\{\hat{\mathbf{A}}_c^{(1)}, \ldots, \hat{\mathbf{A}}_c^{(N)}\} \leftarrow \text{CP-AMG-mult}(\hat{\mathcal{Z}}_c, \tilde{\mathbf{A}}_c^{(1)}, \ldots, \tilde{\mathbf{A}}_c^{(N)}, \text{coarse TFMs})$$

  Compute the CGC $\mathbf{A}^{(n)}$ for $n = 1, \ldots, N$ according to (4.11)
  Apply ν_2 relaxations to $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$
else
  Apply ν_c relaxations to $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$
end

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

$$\mathbf{A}_{\text{CGC}}^{(n)} = \begin{cases} \hat{\mathbf{P}}^{(n)} \hat{\mathbf{A}}_c^{(n)}, & n \in \mathcal{I}_c \\ \hat{\mathbf{A}}_c^{(n)}, & n \in \mathcal{I}_c' \end{cases} \quad \text{for } n = 1, \ldots, N. \tag{4.11}$$

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 [17, 24, 35]). In practice we find that choosing $I_{n,coarsest} \geq R$ for all n works well.

5. Full approximation scheme additive solve phase. The Full Approximation Scheme (FAS) [7] is the nonlinear analogue of the linear additive correction multigrid method. When applied to linear problems FAS reduces to the usual additive method, and so it is a more general multigrid solver. In this section we describe how FAS 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".

5.1. Coarse-level equations. Recall the finest-level equations (4.1), and suppose we define nonlinear operators $\mathbf{H}^{(1)}, \dots, \mathbf{H}^{(N)}$ such that for any $n \in \{1, \dots, N\}$

$$\mathbf{H}^{(n)} \colon \mathbb{R}^{I_1 \times R} \times \cdots \times \mathbb{R}^{I_N \times R} \to \mathbb{R}^{I_n \times R}, \quad (\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}) \mapsto \mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)} - \mathbf{Z}_{(n)}\mathbf{\Phi}^{(n)}$$

where $\mathbf{\Phi}^{(n)}$ is given in (3.2). Then the fine-level problem can be formulated as a system of nonlinear equations

$$\mathbf{H}(\{\mathbf{A}\}) \coloneqq (\mathbf{H}^{(1)}(\{\mathbf{A}\}), \dots, \mathbf{H}^{(N)}(\{\mathbf{A}\})) = (\mathbf{F}^{(1)}, \dots, \mathbf{F}^{(N)}), \tag{5.1}$$

where $\mathbf{F}^{(n)} = \mathbf{0}$ for $n = 1, \dots, N$ on the finest level. Note that $\{\mathbf{A}\}$ is shorthand for $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}$. In order to apply the full approximation scheme we require a coarse version of (5.1). For each mode n we define the coarse operator

$$\mathbf{H}_c^{(n)}(\{\mathbf{A}_c\}) \coloneqq \mathbf{A}_c^{(n)}\mathbf{\Gamma}_c^{(n)} - \hat{\mathbf{Z}}_{(n)}^c\Big(\mathbf{A}_c^{(N)} \odot \cdots \odot \mathbf{A}_c^{(n+1)} \odot \mathbf{A}_c^{(n-1)} \odot \cdots \odot \mathbf{A}_c^{(1)}\Big), \tag{5.2}$$

where $\hat{\mathcal{Z}}_c$ is the coarse-level tensor computed in the multiplicative setup phase. Then the coarse-level FAS equations are given by

$$\mathbf{H}_c(\{\mathbf{A}_c\}) \coloneqq (\mathbf{H}_c^{(1)}(\{\mathbf{A}_c\}), \dots, \mathbf{H}_c^{(N)}(\{\mathbf{A}_c\})) = (\mathbf{F}_c^{(1)}, \dots, \mathbf{F}_c^{(N)}) \tag{5.3}$$

where

$$\mathbf{F}_c^{(n)} = \hat{\mathbf{R}}^{(n)}(\mathbf{F}^{(n)} - \mathbf{H}^{(n)}(\{\mathbf{A}\})) + \mathbf{H}_c^{(n)}(\{\tilde{\mathbf{A}}_c\}) \quad \text{for } n = 1, \dots, N, \tag{5.4}$$

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 (5.3) for $\{\mathbf{A}_c\}$, the coarse-grid-corrected approximations on the fine level are given by

$$\mathbf{A}_{\text{CGC}}^{(n)} = \mathbf{A}^{(n)} + \hat{\mathbf{P}}^{(n)}(\mathbf{A}_c^{(n)} - \tilde{\mathbf{A}}_c^{(n)}) \quad \text{for } n = 1, \dots, N, \tag{5.5}$$

where $\hat{\mathbf{P}}^{(n)}$ is the mode n interpolation operator from the multiplicative setup phase. Together, (5.1) to (5.5) describe a FAS two-level coarse-grid correction scheme for the CP optimality equations. In the following sections we describe the relaxation scheme used in the solve phase, and give a pseudocode description of the multilevel CP-FAS algorithm.

5.2. Relaxation. We employ block nonlinear Gauss–Seidel (BNGS) as the relaxation scheme and coarsest-level solver for the CP-FAS algorithm (Algorithm 2). Applying BNGS to the equations in (5.1) 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

$$\mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)} = \mathbf{Z}_{(n)}\mathbf{\Phi}^{(n)} + \mathbf{F}^{(n)}. \tag{5.6}$$

When considering how to solve (5.6) for mode n, on any level, we recall that an exact solution of the CP optimality equations is a fixed point of FAS if it is a fixed point of the relaxation scheme and the coarsest-level solver. Suppose we update $\mathbf{A}^{(n)}$ by post-multiplying the right-hand side of (5.6) 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 post-multiplying 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 [18] 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 (5.6). 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 (5.6), and that further iterations do little to improve the relaxed approximation. In this paper we use ten GS iterations.

Due to the structure of the FAS equations, in particular the right-hand side in (5.1), 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 §3) are performed only on the finest level.

Algorithm 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 ν_1 relaxations to $\mathbf{H}(\{\mathbf{A}\}) = (\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(N)})$
2. Coarse initial guess:

$$\tilde{\mathbf{A}}_c^{(n)} \leftarrow \begin{cases} \hat{\mathbf{R}}^{(n)}\mathbf{A}^{(n)}, & n \in \mathcal{I}_c \\ \mathbf{A}^{(n)}, & n \in \mathcal{I}'_c \end{cases} \qquad n = 1, \ldots, N$$

3. Coarse right-hand side:

$$\mathbf{F}_c^{(n)} \leftarrow \begin{cases} \mathbf{H}_c^{(n)}(\{\tilde{\mathbf{A}}_c\}) + \hat{\mathbf{R}}^{(n)}(\mathbf{F}^{(n)} - \mathbf{H}^{(n)}(\{\mathbf{A}\})), & n \in \mathcal{I}_c \\ \mathbf{F}^{(n)}, & n \in \mathcal{I}'_c \end{cases} \qquad n = 1, \ldots, N$$

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:

$$\mathbf{A}^{(n)} \leftarrow \mathbf{A}^{(n)} + \begin{cases} \hat{\mathbf{P}}^{(n)}(\mathbf{A}_c^{(n)} - \tilde{\mathbf{A}}_c^{(n)}), & n \in \mathcal{I}_c \\ \mathbf{A}_c^{(n)} - \tilde{\mathbf{A}}_c^{(n)}, & n \in \mathcal{I}'_c \end{cases} \qquad n = 1, \ldots, N$$

6. Apply ν_2 relaxations to $\mathbf{H}(\{\mathbf{A}\}) = (\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(N)})$

else

7. Apply ν_c relaxations to $\mathbf{H}(\{\mathbf{A}\}) = (\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(N)})$

end

5.3. CP-FAS algorithm. A pseudocode description for an additive solve phase V-cycle is given in Algorithm 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 of this additive solution phase and the additive phase in [35]. In the SVD case, singular vectors can be computed in separate V-cycles and FAS is not required since 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 (3.5), which are in some sense equivalent to the singular values, are updated in these FAS cycles as well, making a Ritz step unnecessary.

5.4. 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, i.e., it 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 (FMG) [6, 13, 40]. Full multigrid is based on *nested iterations* whereby coarse levels are used to obtain improved initial guesses for fine-level problems. At any given level the problem is first solved on the next coarser level after which the solution is interpolated to the current level to provide a good initial guess. This process naturally starts at the coarsest level and terminates at the finest. 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 §3). A pseudocode description of the FMG-CP-FAS algorithm is given in Algorithm 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 3 we use a subscript ℓ to index the current level, where $\ell = 0, \ldots, L-1$. Note that a subscript ℓ on an interpolation operator indicates that level ℓ is mapped to level $\ell - 1$.

Algorithm 3: Full multigrid cycle for solve phase of CP decomposition (FMG-CP-FAS)

Output: finest-level factor matrices $\mathbf{A}_0^{(1)}, \ldots, \mathbf{A}_0^{(N)}$

1. On the coarsest level apply ν iterations of ALS with a random initial guess to obtain $\mathbf{A}_{L-1}^{(1)}, \ldots, \mathbf{A}_{L-1}^{(N)}$
2. Set $\ell \leftarrow L-1$
   while $\ell \neq 0$ **do**
3. $\quad \mathbf{A}_{\ell-1}^{(n)} \leftarrow \hat{\mathbf{P}}_\ell^{(n)} \mathbf{A}_\ell^{(n)}$ for $n = 1, \ldots, N$
4. $\quad \{\mathbf{A}_{\ell-1}^{(1)}, \ldots, \mathbf{A}_{\ell-1}^{(N)}\} \leftarrow$ CP-FAS($\{\mathbf{0}\}$, $\mathbf{A}_{\ell-1}^{(1)}, \ldots, \mathbf{A}_{\ell-1}^{(N)}$)
5. $\quad \ell \leftarrow \ell - 1$
   end

6. 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 [2]. 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

$$g(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}) \coloneqq \frac{1}{\|\mathcal{Z}\|}\left(\sum_{n=1}^{N}\|\mathbf{G}^{(n)}\|^2\right)^{1/2}, \tag{6.1}$$

where $\mathbf{G}^{(n)}$ is the mode-n partial derivative of f as defined in (3.1), we iterate until

$$g(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}) < \tau, \tag{6.2}$$

or until the maximum number of iterations are 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 [17, 24, 35], 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 §6.1). 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. These values 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 (6.1).

6.1. 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

$$g(\{\mathbf{A}_{new}\}) > (1-\varepsilon)g(\{\mathbf{A}_{old}\}), \tag{6.3}$$

where the tolerance is set at $\varepsilon = 0.1$. At most five setup cycles are performed, and stopping criterion (6.3) is checked only after three setup cycles have completed. After the setup phase is complete, solution cycles are performed until the stopping criterion (6.2) 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 down-sweep 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 down-sweep 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 down-sweep of CP-AMG-mult. Note that while the TFMs are updated by the down-sweep of CP-AMG-mult, the boot factors are not. We refer to this combination as "Multilevel + FMG" in the tables and figures.

We conclude this section by considering the computational costs of one setup cycle, one solution cycle, and one FMG cycle. Let $\ell = 0, \ldots, L-1$ index the levels, and define

$$I_n^\ell \coloneqq \frac{I_n}{2^\ell}, \qquad P^\ell \coloneqq \prod_{n=1}^{N} I_n^\ell = \frac{1}{2^{N\ell}} \prod_{n=1}^{N} I_n, \qquad S^\ell \coloneqq \sum_{n=1}^{N} I_n^\ell = \frac{1}{2^\ell} \sum_{n=1}^{N} I_n$$

for any $\ell \geq 0$. Assume for simplicity that $\mathcal{Z}$ is dense and that each mode is coarsened at the same rate with L being the same for each mode. Consideration of Algorithm 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 factors 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 $n = 1, \ldots, N$. Computing $\hat{\mathcal{Z}}^c$ 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

$$\left[\left(\frac{2^N}{2^N-1}\right)(n_t(\nu_1+1)+\nu_1+\nu_2+1)+\frac{\nu_c}{2^{N(L-1)}}\right]\cdot\mathcal{O}(NPR) + \left(\frac{2^N}{2^N-1}\right)\cdot\mathcal{O}(PS) \tag{6.4}$$

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 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

$$\left[\left(\frac{2^N}{2^N-1}\right)(\nu_1+\nu_2+1+1/2^N)+\frac{\nu_c}{2^{N(L-1)}}\right]\cdot\mathcal{O}(NPR) \tag{6.5}$$

operations. Similarly, it follows that to leading order one FMG-CP-FAS cycle requires approximately

$$\left[\left(\frac{2^N}{2^N-1}\right)^2(\nu_1+\nu_2+1+1/2^N)+\frac{\nu+(L-1)\nu_c}{2^{N(L-1)}}\right]\cdot\mathcal{O}(NPR) \tag{6.6}$$

operations. Note that in (6.6) ν is the number of ALS iterations performed on the coarsest level and (ν_1, ν_2, ν_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 $\mathcal{Z}$ 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 $\mathcal{Z}$. In our current framework the coarse tensors will in general be dense; multiplication by the inverted Cholesky factors as in (4.5) eliminates any sparsity. Therefore, it may be interesting to consider alternative formulations of the coarse-level equations, for example, working directly with equations of the form in (4.3); see also [35].

TABLE 6.2
Parameters for sparse problem.

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	$N = 8$, $s = 10$, $R = 3$

6.2. 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 $\mathcal{Z}$ of size $s \times s \times \cdots \times s$ with $N = 2d$. We can efficiently construct $\mathcal{Z}$ 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 $[-1, 2, -1]$. While this test problem is somewhat pedagogical in nature, it offers a good starting point to illustrate our

TABLE 6.3

Sparse problem. Average number of iterations and time (in seconds) until the stopping criterion is satisfied with stopping tolerance 10^{-10}. Here 'it' is the number of iterations, 'spd' is the multilevel speedup compared to ALS, 'ns' is the number of successful runs, and 'levs' is the number of levels. 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.

	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

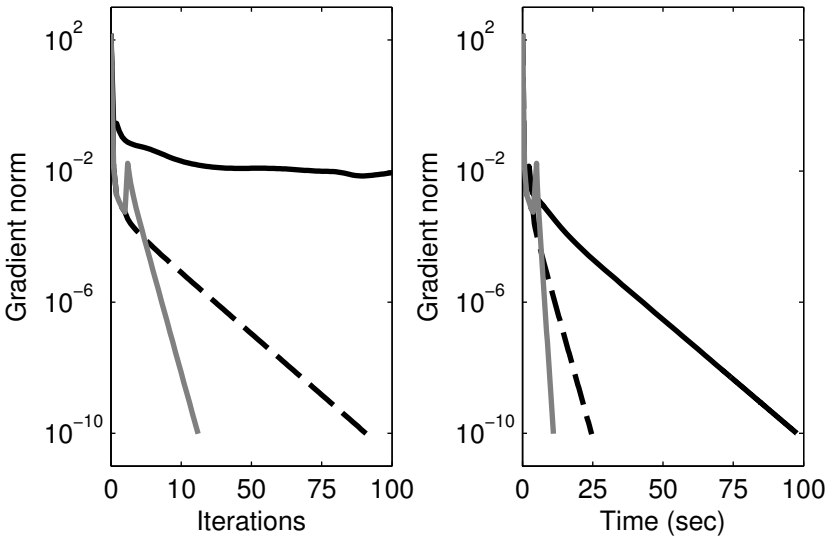

FIG. 6.1. *Sparse problem. Convergence plot for test 3 from Table 6.2 ($N = 4$, $s = 20$, $R = 6$). The solid black line is ALS, the solid gray line is the multilevel method with FMG, and the dashed line is the multilevel method without FMG.*

method. The parameters for the various test tensors that we consider are given in Table 6.2.

The 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 faster than 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)

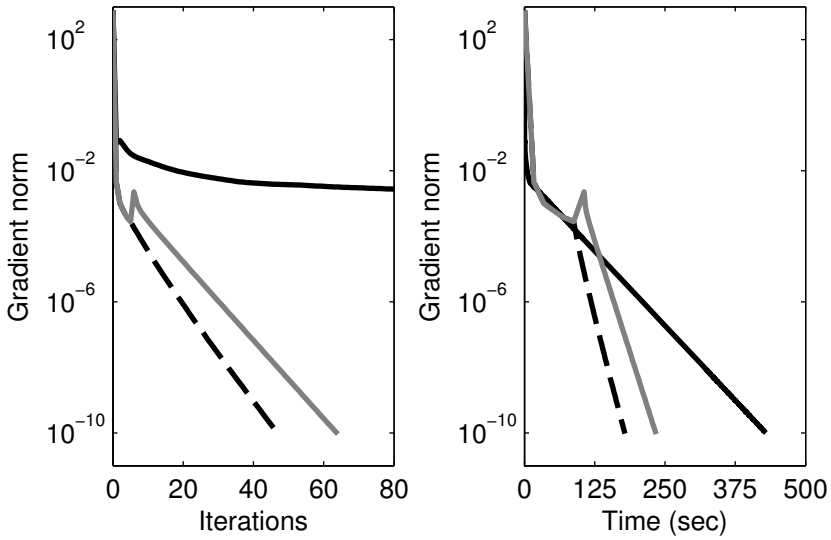

FIG. 6.2. *Sparse problem. Convergence plot for test 9 from Table 6.2 ($N = 6$, $s = 20$, $R = 4$). The solid black line is ALS, the solid gray line is the multilevel method with FMG, and the dashed line is the multilevel method without FMG.*

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 6.1 and 6.2 illustrate the convergence history of ALS and the multilevel method for one run of tests 3 and 9 in Table 6.3, respectively. 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.

6.3. Dense tensor test problem. The second test problem we consider is a dense, symmetric third-order tensor $\mathcal{Z} \in \mathbb{R}^{s \times s \times s}$ whose elements are given by

$$z_{ijk} = \left(i^2 + j^2 + k^2\right)^{-1/2} \quad \text{for } i, j, k = 1, \ldots, s.$$

This tensor was used as a test case in [30], which compares various methods for computing the CP decomposition such as ALS. It was also considered in [31], which describes a novel method for computing the Tucker decomposition of third-order tensors. As mentioned in [31], $\mathcal{Z}$ 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 non-monotonic convergence behavior. The performance of our method when $R \geq 4$ is less robust than desired because the multigrid framework uses

TABLE 6.4

Dense problem. Average number of iterations and time (in seconds) until the stopping criterion is satisfied with stopping tolerance 10^{-10}. Here 'it' is the number of iterations, 'spd' is the multilevel speedup compared to ALS, 'ns' is the number of successful runs, and 'levs' is the number of levels. 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.

		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

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.

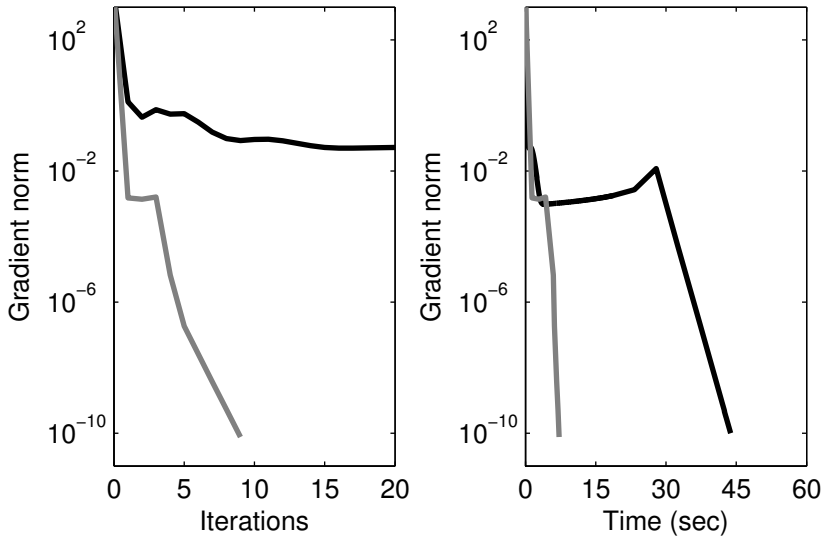

FIG. 6.3. *Dense problem. Convergence plot for test 6 from Table 6.4 ($s = 100$, $R = 3$). The solid black line is ALS and the solid gray line is the multilevel method with FMG.*

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 §6.1). 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

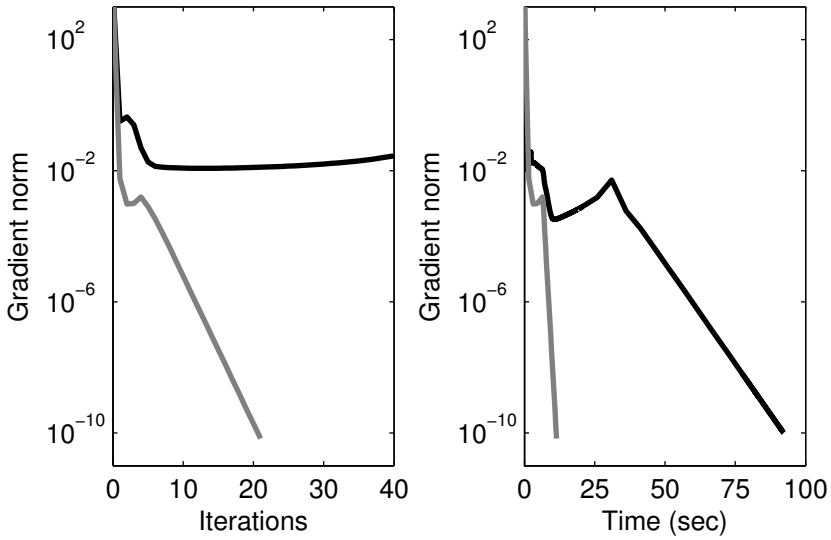

FIG. 6.4. *Dense problem. Convergence plot for test 7 from Table 6.4 ($s = 100$, $R = 4$). The solid black line is ALS and the solid gray line is the multilevel method with FMG.*

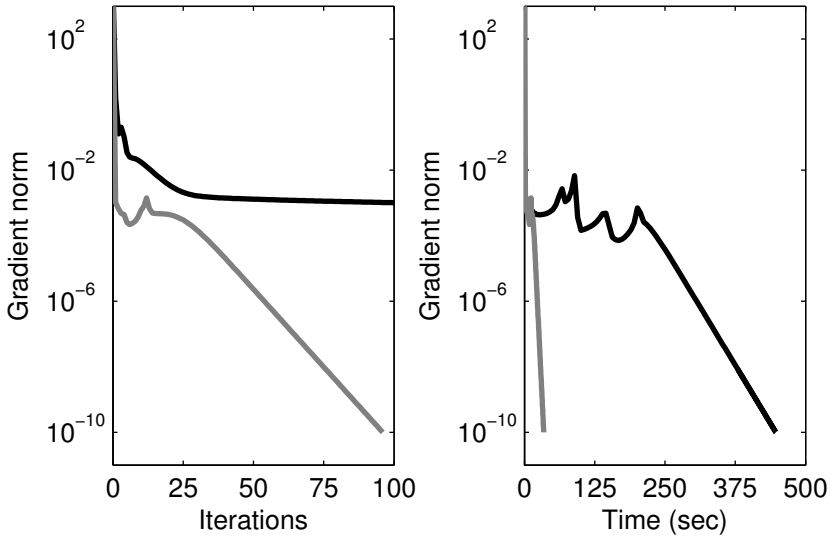

FIG. 6.5. *Dense problem. Convergence plot for test 8 from Table 6.4 ($s = 100$, $R = 5$). The solid black line is ALS and the solid gray line is the multilevel method with FMG.*

and execution time. The speedup is less impressive when $R = 2$, since 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. Such cases were also characterized by slow convergence of ALS.

Figures 6.3, 6.4 and 6.5 illustrate the convergence history of ALS and the multilevel method for one run of tests 6, 7 and 8 in Table 6.4, respectively. 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.

TABLE 6.5

Random data problem. Average number of iterations and time (in seconds) until the stopping criterion is satisfied with stopping tolerance 10^{-10}. Here 'it' is the number of iterations, 'spd' is the multilevel speedup compared to ALS, 'ns' is the number of successful runs, and 'levs' is the number of levels. 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.

		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

6.4. Random data test problem. As our final test problem we consider factoring third-order random data tensors $\mathcal{Z} \in \mathbb{R}^{s \times s \times s}$ 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 factor vectors have a prespecified *collinearity* c. This is a standard non-PDE test problem for CP decomposition that was considered in [1, 38], and also more recently in [34]. Setting the collinearity of the factor vectors to c means that

$$\frac{\left\langle \mathbf{a}_r^{(n)}, \mathbf{a}_q^{(n)} \right\rangle}{\|\mathbf{a}_r^{(n)}\| \cdot \|\mathbf{a}_q^{(n)}\|} = c \quad \text{for } q \neq r \text{ and } r, q = 1, \ldots, R \text{ and } n = 1, \ldots, 3. \tag{6.7}$$

For our test problems we set $c = 0.9$, since it is well-known that collinearity of the factors near unity leads to slow convergence of ALS [38].

The following steps are used to generate a third-order test tensor $\mathcal{Z}$ 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, i.e.,
$$\tilde{\mathbf{A}}^{(n)} = \mathbf{Q}^{(n)}\mathbf{R}^{(n)} \quad \text{for } n = 1, 2, 3.$$
3. Set $\mathbf{A}^{(n)} = \mathbf{Q}^{(n)}\mathbf{L}$ for $n = 1, 2, 3$, and let $\mathcal{Z} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}]\!]$.

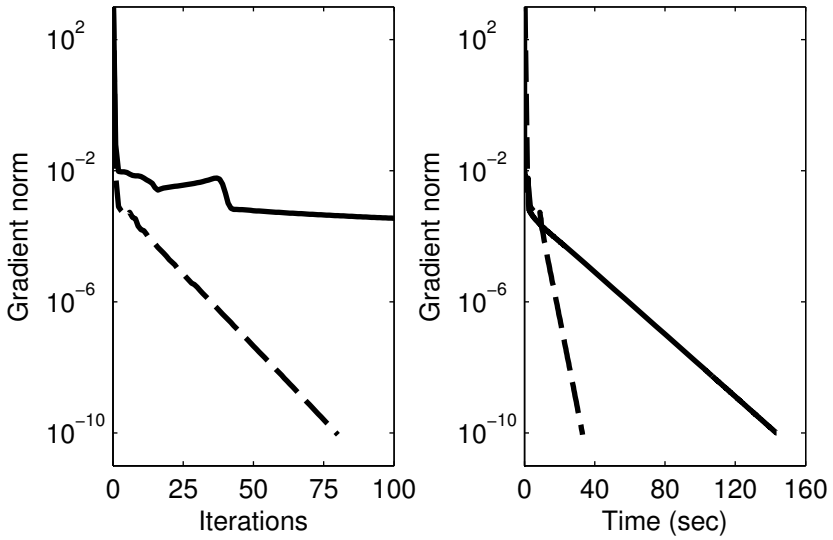

FIG. 6.6. *Random data problem. Convergence plot for test 6 from Table 6.5 ($s = 100$, $R = 5$). The solid black line is ALS and the dashed line is the multilevel method without FMG.*

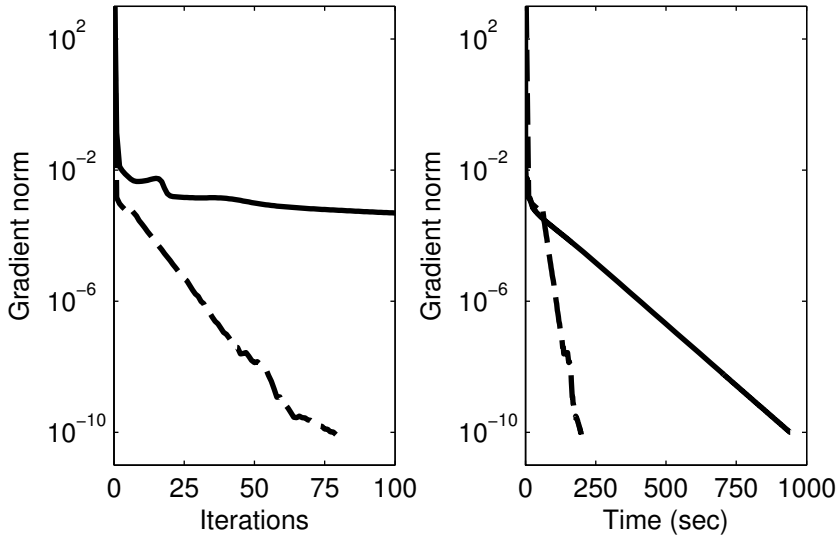

FIG. 6.7. *Random data problem. Convergence plot for test 9 from Table 6.5 ($s = 200$, $R = 4$). The solid black line is ALS and the dashed line is the multilevel method without FMG.*

We note that in [1, 38] 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 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, since 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 grid becomes a coarse grid point), and fine grid points interpolate from their lexicographically nearest neighbors. This coarsening is somewhat arbitrary, since 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 [35], 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 very 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 in Table 6.5, respectively.

7. Concluding remarks. We have presented a new algorithm for computing the rank-R canonical decomposition of a tensor for small R. As far as we are aware, our method is the first genuine multigrid algorithm for computing the CP decomposition. Our work is also significant in that it presents the first adaptive AMG method for a nonlinear optimization problem. Similar to the multilevel method in [35] for computing SVD triplets of a matrix, we combined an adaptive multiplicative setup phase with an additive solve phase. The ALS method was used as the relaxation scheme. Numerical tests with dense and sparse tensors of varying sizes and orders (up to order 8) 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.

Avenues of further research are plentiful. For example, it may be worthwhile to investigate a more sophisticated setup phase that iteratively combines the CP-AMG-mult cycles and CP-FAS-FMG cycles. As discussed briefly in §6.1, an alternative formulation of the coarse-level equations without the inverted Cholesky factors may be fruitful for sparse problems. In addition to PDE-related tensors, there may be other classes of tensors for which multigrid acceleration of ALS may be beneficial, but identifying and studying such classes remains a topic of future research. It would also be interesting to consider other ALS-type methods for the relaxations, for example, ALS accelerated by line search [32], as well as to apply our method to the regularized optimization formulation of CP as described in [1]. Similarly, it would be interesting to generalize our multilevel framework to other similar tensor optimization problems such as the Tucker decomposition [23], block tensor decompositions [26, 29], best rank-$(R_1, \ldots, R_N)$ approximations [16, 27], and to other nonlinear optimization problems.

REFERENCES

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda, *A scalable optimization approach for fitting canonical tensor decompositions*, J. Chemometrics, 25 (2011), pp. 67–86.

[2] B. W. Bader and T. G. Kolda, *Tensor toolbox for Matlab, version 2.4*. last accessed June, 2010.

[3] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, Numer. Linear Algebra Appl., (2012). doi: 10.1002/nla.1818.

[4] M. Bolten, A. Brandt, J. Brannick, A. Frommer, K. Kahl, and I. Livshits, *A Bootstrap Algebraic Multilevel Method for Markov Chains*, SIAM J. Sci. Comput., 33 (2011), pp. 3425–3446.
[5] S. Börm and R. Hiptmair, *Analysis of tensor product multigrid*, Numer. Algorithms, 26 (2001), pp. 219–234.
[6] A. Borzì and G. Borzì, *Algebraic multigrid methods for solving generalized eigenvalue problems*, Int. J. Numer. Meth. Engng., 65 (2006), pp. 1186–1196.
[7] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.
[8] ———, *Multiscale scientific computation: review 2000*, in Multiscale and Multiresolution Methods: Theory and Applications, Springer-Verlag, Berlin, 2001, pp. 1–96.
[9] A. Brandt, J. Brannick, K. Kahl, and I. Livshits, *Bootstrap AMG*, SIAM J. Sci. Comput., 33 (2011), pp. 612–632.
[10] A. Brandt, S. F. McCormick, and J. Ruge, *Multilevel methods for differential eigenproblems*, SIAM J. Sci. Stat. Comput., 4 (1983), pp. 244–260.
[11] M. Brezina, R. D. Falgout, S. MacLachlan, T. A. Manteuffel, S. F. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA) multigrid*, SIAM J. Sci. Comput., 25 (2005), pp. 317–346.
[12] ———, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.
[13] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2000.
[14] 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 (1970), pp. 283–319.
[15] R. A. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis*, UCLA working papers in phonetics, 16 (1970), pp. 1–84.
[16] M. Ishteva, *Numerical methods for the best low multilinear rank approximation of higher-order tensors*, PhD thesis, K. U. Leuven, 2009.
[17] K. Kahl, *Adaptive Algebraic Multigrid for Lattice QCD Computations*, PhD thesis, University of Wuppertal, 2009.
[18] H. B. Keller, *On the solution of singular and semidefinite linear systems by iteration*, J. SIAM Numer. Anal. Ser. B, 2 (1965), pp. 281–290.
[19] B. N. Khoromskij, *On tensor approximation of Green iterations for Kohn-Sham equations*, Comput. Visual. Sci., 11 (2008), pp. 259–271.
[20] ———, *Tensor-structured preconditioners and approximate inverse of elliptic operators in $\mathbb{R}^d$*, Constr. Approx., 30 (2009), pp. 599–620.
[21] B. N. Khoromskij and V. Khoromskaia, *Multigrid accelerated tensor approximation of function related multidimensional arrays*, SIAM J. Sci. Comput., 31 (2009), pp. 3002–3026.
[22] T. G. Kolda, *Multilinear operators for higher-order decompositions*, Tech. Report SAND2006-2081, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California, 2006.
[23] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[24] D. Kushnir, M. Galun, and A. Brandt, *Efficient multilevel eigensolvers with applications to data analysis tasks*, IEEE Trans. Pattern Anal. Mach. Intell., 32 (2010), pp. 1377–1391.
[25] L. De Lathauwer, *A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 642–666.
[26] ———, *Decompositions of a higher-order tensor in block terms – part II: Definitions and uniqueness*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1033–1066.
[27] 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 (2000), pp. 1324–1342.
[28] ———, *Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 295–327.
[29] 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 (2008), pp. 1067–1083.
[30] I. V. Oseledets and D. V. Savost'yanov, *Minimization methods for approximating tensors and their comparison*, Comp. Math. Math. Phys., 46 (2006), pp. 1641–1650.
[31] 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

(2008), pp. 939–956.
[32] M. Rajih, P. Comon, and R. A. Harshman, *Enhanced line search: a novel method to accelerate PARAFAC*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1128–1147.
[33] V. De Silva and Lek-Heng Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM. J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127.
[34] H. De Sterck, *A nonlinear GMRES optimization algorithm for canonical tensor decomposition*, (2012). SIAM J. Sci. Comput., accepted, arXiv: 1105.5331.
[35] ———, *A self-learning algebraic multigrid method for extremal singular triplets and eigenpairs*, (2012). SIAM J. Sci. Comput., accepted, arXiv: 1102.0919.
[36] H. De Sterck, K. Miller, E. Treister, and I. Yavneh, *Fast multilevel methods for Markov chains*, Numer. Linear Algebra Appl., (2011). doi: 10.1002/nla.800.
[37] G. Styan, *Hadamard products and multivariate statistical analysis*, Linear Algebra Appl., 6 (1973), pp. 217–240.
[38] G. Tomasi and R. Bro, *A comparison of algorithms for fitting the PARAFAC model*, Comput. Stat. Data Anal., 50 (2006), pp. 1700–1734.
[39] E. Treister and I. Yavneh, *On-the-fly adaptive smoothed aggregation multigrid for Markov chains*, SIAM J. Sci. Comput., 33 (2011), pp. 2927–2949.
[40] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Elsevier Academic Press, San Diego, CA, 2001.