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

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.

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 $I_1 \times \cdots \times I_N$ [19]. 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 canonical tensor 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, let $\mathcal{T}$ be an arbitrary Nth-order tensor of rank R, meaning that it can be expressed as a sum of no fewer than R rank-one components. Then, the canonical decomposition of $\mathcal{T}$ 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 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 $n = 1, \ldots, N$, the vectors $\mathbf{a}_1^{(n)}, \ldots, \mathbf{a}_R^{(n)}$ can be stored as the columns of an $I_n \times R$ matrix $\mathbf{A}^{(n)}$. 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 [9, 16]. 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 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 in [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. In this paper, we compute (local) minimizers of f by using algebraic multigrid (AMG) techniques to solve the optimality equations directly.

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 [19]. This manifold arises because of a scaling indeterminacy inherent to CP, 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 [19]. In §3 we explain how

*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 (k7miller@uwaterloo.ca).

the scaling and permutation indeterminacies are removed by imposing a specific normalization and ordering of the factors. However, CP 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 [13]. Uniqueness of an 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 application fields [19].

The primary application of CP 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 [18, 21, 22]. Many algorithms have been proposed for computing the CP decomposition [1, 19, 12, 23], however, the workhorse algorithm is still the original ALS method, first proposed in 1970 in early papers on CP [9, 16]. 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. As a result, ALS-type algorithms are considered 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 multiplicative corrections are used in conjunction with Bootstrap algebraic multigrid (BAMG) interpolation [6, 7] to not only build the necessary transfer operators and coarse-level tensors, but also to compute initial approximations of the factor matrices. The setup phase is adaptive in the sense that the transfer operators are continually improved using the most recent approximation of 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 based on the Full Approximation Scheme (FAS) [5, 8, 25] are used in the solve phase, which can still converge when the exact solution lies only approximately in the range of interpolation. We note that the combination of a multiplicative setup scheme and BAMG was already considered in [7, 20], in the context of multilevel eigensolvers. Our multigrid framework is closely related to recent work on an adaptive AMG solver for extremal singular triplets and eigenpairs of matrices [14], and to a lesser degree to multigrid methods for Markov chains [3, 15, 24].

We expect our method to work well for tensors with properties that make a multilevel approach beneficial, for example, for certain tensors that arise in the context of PDE discretization on high-dimensional regular lattices [18, 21, 22]. 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 computing SVD triplets of a matrix via AMG [14].

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 briefly reviews basic operations and identities that are essential to our paper. Much of our notation has been adopted from [1, 19], and for further details we refer to the survey article by Kolda and Bader [19], 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., $\mathfrak{Z}$. 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 $\mathfrak{Z}$ 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 a matrix row/column, which is obtained by fixing every index of a tensor but one. Given a tensor $\mathcal{Z}$ the mode-n matricized version is denoted by $\mathbf{Z}_{(n)}$.

The *n-mode 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 elegantly expressed in terms of unfolded tensors: $\mathcal{X} = \mathcal{Z} \times_n \mathbf{A} \Leftrightarrow \mathbf{X}_{(n)} = \mathbf{A}\mathbf{Z}_{(n)}$, from which it can be seen that $\mathcal{Z} \times_n \mathbf{A} \times_n \mathbf{B} = \mathcal{Z} \times_n (\mathbf{B}\mathbf{A})$ for any matrices $\mathbf{A}$ and $\mathbf{B}$ of the appropriate sizes. Another important property that we make use of extensively is as follows. Let $\mathbf{A}^{(n)} \in \mathbb{R}^{J_n \times I_n}$ for $n = 1, \ldots, N$. Then, for any $n \in \{1, \ldots, N\}$

$$\mathcal{X} = \mathcal{Z} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)} \quad \Leftrightarrow \quad \mathbf{X}_{(n)} = \mathbf{A}^{(n)} \mathbf{Z}_{(n)} \left(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(1)} \right)^T, \tag{2.1}$$

where $\otimes$ is the Kronecker product.

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 $\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \cdots \mathbf{a}_K \otimes \mathbf{b}_K]$. The Khatri-Rao and Kronecker products have many useful properties, however, we only need the associativity of the Khatri-Rao product, and the *mixed-product property* of the Kronecker product, i.e., $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{A}\mathbf{C} \otimes \mathbf{B}\mathbf{D}$. It is then easy to prove the following useful identity,

$$\mathbf{A}^{(1)}\mathbf{B}^{(1)} \odot \cdots \odot \mathbf{A}^{(N)}\mathbf{B}^{(N)} = \left(\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)} \right) \left(\mathbf{B}^{(1)} \odot \cdots \odot \mathbf{B}^{(N)} \right), \tag{2.2}$$

for any sequences of matrices $\mathbf{A}^{(n)}$ and $\mathbf{B}^{(n)}$, $n = 1, \ldots, N$, of the appropriate sizes.

3. CP first-order optimality equations and alternating least squares. The CP first-order optimality equations are obtained by setting the gradient of the functional in (1.2) equal to zero. Following the derivation in [1], for each mode $n \in \{1, \ldots, N\}$ the partial 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 $n = 1, \ldots, N$. The first-order optimality equations are then given by $\mathbf{G}^{(n)} = \mathbf{0}$ for $n = 1, \ldots, N$. 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. Moreover, if each $\mathbf{A}^{(n)}$ has full rank then $\mathbf{\Gamma}^{(n)}$ will be symmetric positive definite (SPD).

One iteration of ALS is equivalent to one iteration of block nonlinear Gauss–Seidel (BNGS) applied to the optimality equations. 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 in 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

$$\mathbf{a}_r^{(n)} \mapsto \lambda_r \left(\mathbf{a}_r^{(n)} / \|\mathbf{a}_r^{(n)}\| \right) \quad \text{for } n = 1, \ldots, N, \quad \lambda_r = \left(\|\mathbf{a}_r^{(1)}\| \ldots \|\mathbf{a}_r^{(N)}\| \right)^{1/N} \tag{3.4}$$

for $r = 1, \ldots, R$. This normalization equilibrates the norms of the factors of each component, i.e., $\|\mathbf{a}_r^{(1)}\| = \cdots = \|\mathbf{a}_r^{(N)}\|$ for $r = 1, \ldots, R$. 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. We employ two-level notation whereby 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.

4.1. Derivation of coarse-level equations. The fine-level equations are given by the gradient equations stated in §3,

$$\mathbf{Z}_{(n)}\Big(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}\Big) = \mathbf{A}^{(n)}\mathbf{\Gamma}^{(n)} \quad \text{for } n = 1, \ldots, N. \tag{4.1}$$

Suppose there exist N full rank operators $\mathbf{P}^{(n)} \in \mathbb{R}^{I_n \times I_{n,c}}$ with $I_{n,c} < I_n$, 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 we can achieve equality). 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 (2.1) and (2.2), 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, and 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 each mode n, and again appealing to properties (2.1) and (2.2), 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

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

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 conditions are satisfied only approximately, we expect (4.7) to yield an improved but not exact fine-level solution.

4.2. Bootstrap AMG V-cycles. This section describes how we use Bootstrap AMG to compute initial approximations of the desired factor matrices, and to adaptively determine the interpolation operators that approximately fit the factor matrices. We follow the approaches outlined in [7, 20, 14].

We begin by describing the initial BAMG V-cycle. On the finest level we randomly choose n_t *test blocks*, where each test block is a collection of N randomly generated *test factor matrices* (TFMs) $\mathbf{A}_t^{(1)}, \ldots, \mathbf{A}_t^{(N)}$. The reason we must consider test blocks instead of simply adding more columns to the factor matrices, is that rank-one components of the best rank-R CP tensor approximation must be found simultaneously [19]; contrary to the best rank-R matrix approximation, one cannot obtain the best rank-R CP approximation 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 we relax on each of the test blocks separately and also on the BFMs using the ALS algorithm described in §3. We then coarsen each mode on the finest level and determine the interpolation operators $\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(N)}$. 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 $\mathbf{P}^{(n)}$. We also build the coarse-level tensor $\hat{\mathcal{Z}}^c$ and restrict the TFMs and BFMs to the first coarse level. This process is then repeated recursively until some coarsest level is reached, from which point on we relax only on the BFMs. On the coarsest level an approximate solve for the BFMs is performed via ALS.

In the upward sweep of the first cycle, starting from the coarsest level, the BFMs are interpolated to the next finer level which gives the coarse-grid correction (CGC) on that level. The CGC is then relaxed by 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 our 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 in the second phase 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 is appropriate 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. While the simple coarsening procedure discussed here works well for the test problems considered in §6, more general coarsening algorithms for other types of tensors would be desirable. The development of such algorithms remains an interesting topic of future research.

Since each mode is coarsened independently of any other mode, it is necessary to define a mode-dependent threshold value $I_{n,coarsest}$ as the maximum size of the coarsest level for each mode n. Then, at any level, any mode that has not yet reached its respective threshold value is coarsened further. As a result, on coarser levels it may be possible to perform restrictions/interpolations over a subset of the modes. For example, the coarse-level tensor may be obtained by taking the product in (4.5) only over the modes that require further coarsening. We note that the size of the coarsest level is crucial to the performance our method, and, in particular, too small a threshold may negatively impact convergence, or may even cause divergence (as in [20, 14]). In practice we find that choosing $I_{n,coarsest} \geq R$ for all n works well.

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 $\mathbf{U}_b = \mathbf{A}_b^{(n)}$. Following the approaches of [6, 20, 14] 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 of $\mathbf{P}^{(n)}$ that corresponds to a point in $\mathcal{F}_n$ may now be determined consecutively by independent least squares fits. Consider a fixed point $i \in \mathcal{F}_n$ with coarse interpolatory set $\mathcal{C}_n^i$. The following least squares problem is solved to determine 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, i.e., $|\mathcal{C}_n^i| \leq M_s$. 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

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

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 diagonal weight matrix $\mathbf{M} = \text{diag}(\boldsymbol{\mu})$ are solved by a standard normal equations approach.

5. Full approximation scheme additive solve phase. The Full Approximation Scheme (FAS) [5] 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. As in §4 coarse-level quantities denoted by a subscript "c".

5.1. Coarse-level equations. Recall the finest-level equations in the multiplicative setup phase (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 by (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 we use $\{\mathbf{A}\}$ as shorthand for $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}$. In order to apply FAS 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.

5.2. Relaxation. We employ block nonlinear Gauss–Seidel (BNGS) as the relaxation scheme and coarsest-level solver for the FAS solution cycles. 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 observe 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 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 [17] for positive semidefinite matrices implies 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 implicit assumption that the factor matrices have nonzero columns. 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.

5.3. Full multigrid FAS cycles. For some tensors the initial guess provided by the multiplicative setup phase may be inadequate to yield a convergent solve phase, i.e., it may lie 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) [4, 25]. Full multigrid is based on the idea of *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 (approximately) on the next coarser level and 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 FAS solution cycles to obtain an accurate solution. We use one FAS V-cycle as the solver on each level of the FMG cycle, except on the coarsest level where 200 iterations of ALS are used (see §3).

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, with

$$g(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}) = \left\| \left[{\mathbf{G}^{(1)}}^T \mid \cdots \mid {\mathbf{G}^{(N)}}^T \right] \right\| / \|\mathcal{Z}\|, \tag{6.1}$$

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 $\tau = 10^{-10}$. In the setup phase we use V(5, 5)-cycles with 100 relaxations on the coarsest level and $n_t = 2$ test blocks. In the solve phase we use V(1, 1)-cycles with 50 relaxations on the coarsest level. As in [20, 14], a larger number of relaxations is required in the setup cycles to produce sufficiently accurate transfer operators.

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 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.

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 stagnation criterion (6.3) is checked only after three setup cycles have elapsed. Once 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 are rebuilt by one down-sweep of a setup phase V-cycle with the previous iterate $\{\mathbf{A}_{old}\}$ input for the boot factors. Note that the boot factors are not updated

by the down-sweep, as doing so would likely ruin any progress made by the solution cycles. The solve phase stagnation check is performed at most once, and any further indications of stagnation are ignored.

The combination of the setup and solve phases described above can be modified to include FMG as part of the setup phase. After the setup cycles have completed, we perform one FMG cycle to compute a new approximation to the boot factors. The transfer operators are then rebuilt using one down-sweep of a setup phase V-cycle. Note that while the TFMs are updated by the down-sweep, the boot factors are not. We refer to this combination as "Multilevel + FMG" in the tables.

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 method.

TABLE 6.1

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.

		ALS			Multilevel				Multilevel + FMG				
test	problem parameters	it	time	ns	it	time	spd	ns	it	time	spd	ns	levs
1	$N = 4$, $s = 20$, $R = 4$	1897	24.0	10	37	7.6	3.2	10	36	8.0	3.1	10	2
2	$N = 4$, $s = 20$, $R = 5$	3329	53.6	8	64	15.1	4.5	10	42	11.8	5.1	10	2
3	$N = 4$, $s = 20$, $R = 6$	3587	70.3	9	67	17.5	4.0	9	32	10.6	6.8	10	2
4	$N = 4$, $s = 50$, $R = 2$	5457	105.9	10	123	40.5	2.7	10	120	41.2	2.6	10	4
5	$N = 4$, $s = 50$, $R = 3$	5508	150.3	4	182	64.9	2.3	9	99	40.9	3.8	9	4
6	$N = 4$, $s = 50$, $R = 4$	6788	244.0	3	150	62.0	5.2	10	136	58.7	5.4	9	4
7	$N = 6$, $s = 20$, $R = 2$	1619	187.0	10	48	111.4	1.7	10	52	128.3	1.5	10	3
8	$N = 6$, $s = 20$, $R = 3$	3481	610.0	10	72	164.8	3.7	10	70	177.9	3.5	10	3
9	$N = 6$, $s = 20$, $R = 4$	4085	939.5	10	76	209.6	4.5	10	78	228.9	4.2	10	3
10	$N = 8$, $s = 10$, $R = 2$	634	229.5	10	48	170.8	1.3	10	56	203.5	1.1	10	3
11	$N = 8$, $s = 10$, $R = 3$	1743	943.3	10	39	402.3	2.3	10	43	474.0	2.0	10	3

The results in Table 6.1 show that our multilevel approach is anywhere from two to seven times faster than ALS for the sparse 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 6.1, the speedup tends to increase as the number of components R increases.

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 [21] and in [22], which discussed methods for computing the CP and Tucker decompositions, respectively. As mentioned in [22], $\mathcal{Z}$ 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. We compute decompositions of $\mathcal{Z}$ for $R = 2, 3, 4$. 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 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 6.2. We note that only the multilevel method with FMG is considered (see the description in §6.1). For $R \geq 3$ our multilevel approach can lead to significant savings in iterations and execution time. The speedup is less impressive when $R = 2$, since ALS already converges quickly without any multigrid acceleration. 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. We note that it may be possible to improve robustness for challenging problems such as this by devising a more sophisticated setup phase, however, this remains the topic of future work.

TABLE 6.2

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.

		ALS			Multilevel + FMG				
test	problem parameters	it	time	ns	it	time	spd	ns	levs
1	$s = 50$, $R = 2$	161	0.7	10	7	2.2	0.3	10	5
2	$s = 50$, $R = 3$	2435	11.8	10	15	2.7	4.7	10	5
3	$s = 50$, $R = 4$	4838	26.1	5	97	10.1	4.4	7	4
4	$s = 100$, $R = 2$	253	10.7	10	7	7.9	1.4	10	6
5	$s = 100$, $R = 3$	1695	80.2	9	9	8.1	10.1	10	6
6	$s = 100$, $R = 4$	3836	202.2	6	82	27.5	14.1	9	5
7	$s = 200$, $R = 2$	274	90.3	10	7	48.8	1.9	10	7
8	$s = 200$, $R = 3$	1830	682.3	10	12	61.7	11.2	10	7
9	$s = 200$, $R = 4$	2998	1249.5	8	79	178.0	11.6	9	6

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 [14] for computing SVD triplets of a matrix, we combined an adaptive multiplicative setup phase with an additive solve phase. 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.

Avenues of further research are plentiful. For example, it may be worthwhile to investigate a more sophisticated setup phase that iteratively combines setup and solve cycles along with FMG. 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; identifying and studying such classes remains an open topic of research. It would also be interesting to consider other ALS-type methods to play the role of the relaxation scheme, for example, ALS with a line search after each major iteration. It may also be possible to generalize our multilevel framework to other similar tensor optimization problems such as the regularized optimization formulation of CP as described in [1], the Tucker decomposition [19], block tensor decompositions [10], best rank-$(R_1, \ldots, R_N)$ approximations [11], 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] 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., (2010). accepted for publication.

[4] A. Borzì and G. Borzì, *Algebraic multigrid methods for solving generalized eigenvalue problems*, Int. J. Numer. Meth. Engng., 65 (2006), pp. 1186–1196.

[5] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[6] ———, *Multiscale scientific computation: review 2000*, in Multiscale and Multiresolution Methods: Theory and Applications, Springer-Verlag, Berlin, 2001, pp. 1–96.

[7] A. Brandt, J. Brannick, K. Kahl, and I. Livshits, *Bootstrap AMG*, SIAM J. Sci. Comput., 33 (2011), pp. 612–632.

[8] A. Brandt, S. F. McCormick, and J. Ruge, *Multilevel methods for differential eigenproblems*, SIAM J. Sci. Stat. Comput., 4 (1983), pp. 244–260.

[9] 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.

[10] L. De Lathauwer, *Decompositions of a higher-order tensor in block terms – part II: Definitions and uniqueness*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1033–1066.

[11] 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.

[12] ———, *Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 295–327.

[13] 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 (2008), pp. 1084–1127.

[14] H. De Sterck, *A self-learning algebraic multigrid method for extremal singular triplets and eigenpairs*, (2011). Submitted to SIAM J. Sci. Comput., arXiv: 1102.0919.

[15] H. De Sterck, K. Miller, E. Treister, and I. Yavneh, *Fast multilevel methods for Markov chains*, Numer. Linear Algebra Appl., 18 (2011), pp. 961–980.

[16] 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.

[17] 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.

[18] B. Khoromskij, *Tensor-structured preconditioners and approximate inverse of elliptic operators in $\mathbb{R}^d$*, Constr. Approx., 30 (2009), pp. 599–620.

[19] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[20] 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.

[21] 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.

[22] 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.

[23] G. Tomasi and R. Bro, *A comparison of algorithms for fitting the PARAFAC model*, Comput. Stat. Data Anal., 50 (2006), pp. 1700–1734.

[24] E. Treister and I. Yavneh, *On-the-fly adaptive smoothed aggregation multigrid for Markov chains*, SIAM J. Sci. Comput., 33 (2011), pp. 2927–2949.

[25] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Elsevier Academic Press, San Diego, CA, 2001.